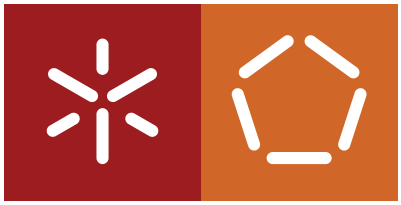


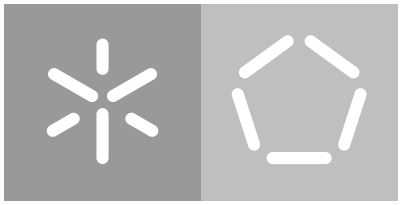
Universidade do Minho

Escola de Engenharia

Ana Soraia Martins

Aplicação web para Corpus de Interpretação Multimédia Bidireccional (português-inglês) do Parlamento Europeu

Outubro de 2017



Universidade do Minho

Escola de Engenharia

Ana Soraia Martins

Aplicação web para Corpus de Interpretação Multimédia Bidireccional (português-inglês) do Parlamento Europeu

Dissertação de Mestrado

Mestrado Integrado em Engenharia de Telecomunicações e Informática

Trabalho efetuado sob a orientação de

Professor Doutor Sérgio Adriano Fernandes Lopes

Professora Doutora Sílvia Lima Gonçalves Araújo

Outubro de 2017

Abstract

interPE is a web application developed to host a Portuguese-English multimedia interpretation *corpus* of European Parliament speeches, with the purpose to present a simple and user-friendly interface.

Even though there are many *corpora* in text format, there are very little multimedia interpretation *corpora* known, being that video and audio are complementary for a better understandig and analysis of the speech. One of the greatest limitations when using currently existing multimida *corpora* is the fact that they have very complicated and not user friendly interfaces, which makes their use complex. Besides, portuguese rarely is present and there isn't any bidirectional portuguese multimedia *corpus*.

The app provides two types of search: simple and avdvanced search (by the speeches metadata). For both situations, the search is bilingual, which means that for each result with the searched query on the selected language, the corresponding word or expression on the other language is displayed. Besides, the app exhibites results for original and interpreted source speeches on the selected language.

After displaying the results, it's possible to watch the selected speech with synchronized subtitles. Appart from watching the speech, the user also can check the metadata information of a speech and, also, just read the transcriptions from the speech.

In addition to the search feature, *interPE* offers the ability of adding new speeches to the Data Base, making the application always updated and in constant growth.

Between the technologies used on the development of the *interPE* app are *Vue.js* for the *frontend*, *Node.js* for the *backend* and *MongoDB* for the Data Base.

Resumo

O *interPE* é uma aplicação *web* desenvolvida para albergar o *corpus* de interpretação multimédia Português-Inglês de discursos do Parlamento Europeu, com o intuito de apresentar uma interface simples e intuitiva.

Apesar da existência de vários *corpora* em formato de texto, poucos são os *corpora* de interpretação multimédia conhecidos, sendo que o vídeo e áudio servem de complemento para uma melhor compreensão e análise da fala ou discursos. Uma das maiores limitações de utilização dos *corpora* multimédia actualmente existentes é o facto das suas interfaces serem complexas e pouco intuitivas, dificultando o seu manuseamento. Para além disso, o português é uma língua raramente presente, sendo que não existe nenhum *corpus* multimédia português bidireccional.

O *interPE* permite a pesquisa simples ou avançada (por metadados) de palavras ou expressões presentes nos discursos. Em ambas as situações, a pesquisa é bilingue, ou seja, para cada resultado apresentado com a palavra ou expressão pesquisada é exibido o correspondente segmento na outra língua. Para além disso, a aplicação apresenta os resultados de pesquisa tanto para os discursos originais como interpretados, na língua de pesquisa seleccionada.

Após a listagem de resultados, é possível visualizar o discurso seleccionado com as respectivas legendas bilingue sincronizadas, além disso, é possível consultar os metadados do discurso, assim como consultar apenas as transcrições do mesmo em texto.

Para além da típica funcionalidade de pesquisa, o *interPE* oferece ainda a possibilidade de adicionar novos discursos à Base de Dados, tornando possível o constante crescimento e enriquecimento deste *corpus*.

Entre as tecnologias utilizadas no desenvolvimento do *interPE* estão o *Vue.js* para o *frontend*, *Node.js* para o *backend* e ainda, *MongoDB* ao nível da Base de Dados.

Conteúdo

1 Introdução **1**

1.1 Enquadramento 1

1.2 Motivação 2

1.3 Objectivos 2

1.4 Estrutura da Dissertação 2

2 Estado da Arte **4**

2.1 TCSE 4

2.2 Veiga 6

2.3 EPIC 8

2.4 Análise Comparativa 9

2.5 Tecnologias *web* 9

3 Análise do Sistema **12**

3.1 Requisitos do Sistema 12

3.2 Casos de Uso 13

3.3 Arquitectura do Sistema 15

3.4 Concepção da Interface 16

3.4.1 Navegação 16

3.4.2 Esboços da Interface 17

3.5 Tecnologias a utilizar 22

3.5.1 Escolha *Front-end Framework* 22

3.5.2 Escolha da Base de Dados 24

4 Desenvolvimento do projecto **27**

4.1 Estrutura da Base de Dados 27

4.2 Cliente 28
4.2.1 Componentes 29
4.2.2 Pesquisar no InterPE 34
4.2.3 Visualização dos Discursos 37
4.2.4 Adição de Discursos 41
4.3 Servidor 44
4.3.1 Estrutura Servidor 44
4.3.2 Pesquisar e Visualizar Discursos no *interPE* 45
4.3.3 Adição de Discursos 47

5 Conclusão **48**
5.1 Resultados 48
5.2 Conclusões 48
5.3 Trabalho Futuro 49

Referências **54**

Lista de Figuras

2.1 Página de resultados do TCSE 5
2.2 Página de visualização de um discurso no TCSE 6
2.3 Página de pesquisa do *corpus Veiga* 7
2.4 Lista de resultados do *corpus Veiga* após a pesquisa da palavra galega "*axuda*" 7
2.5 Visualização de discursos do *corpus Veiga* 8

3.1 Diagrama Casos de Uso Página principal 14
3.2 Diagrama Casos de Uso Visualização de Resultados 15
3.3 Diagrama Casos de Uso Adição Novo Discurso 15
3.4 Arquitectura do *interPE* 16
3.5 Diagrama de Navegação do *interPE* 17
3.6 Diagrama dos diferentes estados da *home page* 17
3.7 Esboço *Home Page* do *interPE* 18
3.8 Esboço visualização de resultados do *interPE* 19
3.9 Esboço visualização de um discurso do *interPE* 20
3.10 Esboço página de Contactos e *About* do *interPE* 21
3.11 Esboço página de adição de discursos do *interPE* 22
3.12 Ficheiro simples vue.js 23
3.13 Estrutura da lista de compras utilizando SQL 25
3.14 Estrutura da lista de compras utilizando NoSQL 25

4.1 Componente *Search.vue* 30
4.2 Componente *Results.vue* 30
4.3 Componente *VideoPlayer.vue* 31
4.4 Componente *Text.vue* 31
4.5 Componente *AddSpeech.vue* 32

4.6 Componente *Home.vue* 32
4.7 Componente *Nav.vue* 33
4.8 Componente *App.vue* 33
4.9 Hierarquia dos componentes do *interPE* 33
4.10 Opções de pesquisa avançada no *interPE* 35
4.11 Consulta de metadados de um discurso no *interPE* 36
4.12 Diagrama de sequência de uma pesquisa no *interPE* 37
4.13 Estrutura de um ficheiro '.srt' 38
4.14 Estrutura de um ficheiro '.exb' 39
4.15 Diagrama de sequência da função de visualização de discursos do *interPE* . . 41
4.16 Formulário de login do *interPE* 43
4.17 *Modal* de ajuda na adição de um novo discurso no *interPE* 43
4.18 Diagrama de sequência da função de adição de discursos do *interPE* 44

1 Introdução

1.1 Enquadramento

Um *corpus*, tal como referido no Dicionário Priberam da Língua Portuguesa [1], é um conjunto de documentos que servem de base para a descrição ou estudo de um fenómeno. No caso de um *corpus* linguístico, este é constituído por uma colecção de textos, transcrições de discursos, cujo vocabulário é representativo da língua.

Através dos *corpora* linguísticos, é possível analisar vários aspectos importantes da linguagem tais como a frequência de utilização de uma palavra, as diferenças entre um discurso escrito e falado, ou entre um discurso formal e informal, a distribuição de vocabulário entre diferentes línguas, entre outros fenómenos linguísticos.

O *corpus* audiovisual de interpretação simultânea português-inglês para intervenções no parlamento europeu, proposto neste trabalho, o *interPE*, será constituído não por textos mas por vídeos e, também, as suas respectivas transcrições, daí ser denominado por *corpus* audiovisual. Para além disso, este será um *corpus* bidireccional uma vez que o seu conteúdo consistirá em discursos cuja língua original é o inglês, com a respectiva interpretação em português, e em discursos cuja língua original é o portugês, com a respectiva interpretação em inglês.

Além de audiovisual, o *interPE* é um *corpus* de interpretação simultânea o que significa que ao mesmo tempo que o eurodeputado discursa, as suas ideias são exprimidas oralmente noutra língua por um intérprete, sem que o eurodeputado realize pausas no seu discurso para o segundo falar. Os discursos disponibilizados no formato de texto correspondem à transcrição dos discursos interpretados e originais.

O *interPE* é um *corpus* multimédia constituído por uma colecção de vídeos das intervenções dos eurodeputados portugueses e ingleses, em sessão plenária, com a respectiva interpretação para a outra língua e a sua transcrição.

1.2 Motivação

Apesar da existência de vários *corpora* em formato de texto, poucos são os *corpora* de interpretação multimédia conhecidos, sendo que o vídeo e áudio servem de complemento para uma melhor compreensão e análise da fala ou discursos.

Uma das maiores limitações de utilização dos *corpora* multimédia actualmente existentes é o facto das suas interfaces serem complexas e pouco intuitivas, dificultando o seu manuseamento. Para além disso, o português é uma língua raramente presente, sendo que não existe nenhum *corpus* multimédia português bidireccional.

Portanto, a existência de uma interface intuitiva e de um *corpus* audiovisual bidireccional português são grandes motivações para o desenvolvimento deste projecto, sendo que a concretização do mesmo, para além de contribuir para o estudo linguístico, pode também servir de ferramenta para a aprendizagem do português ou inglês, uma vez que a bidireccionalidade do *interPE* permite a comparação entre as duas línguas.

1.3 Objectivos

Os objectivos definidos para o desenvolvimento deste projecto são os seguintes:

1. Elencar as principais limitações e desafios para as ferramentas de acesso a um *corpus* multimédia de interpretação simultânea;

2. Desenvolver uma aplicação web que fornece acesso a um *corpus* multimédia de interpretação simultânea português-inglês com dois níveis de utilização: pesquisa simples (por defeito) e pesquisa avançada em metadados;

3. Verificar se as funcionalidades do HTML5 são suficientes para construir uma *Single-Page Applicaton*;

1.4 Estrutura da Dissertação

Esta dissertação está divida em 5 capítulos que tratam de diferentes temáticas. O capítulo 2, *Estado da Arte*, tem como objectivo descrever as aplicações de *corpora* multimédia multilingues actualmente existentes.

No capítulo 3, é realizada uma *Análise do Sistema*, onde são descritos os requisistos do sistema, assim como a sua arquitectura, casos de uso, estrutura da interface e, ainda, algumas escolhas tomadas para o desenvolvimento da aplicação *web*.

O capítulo 4 trata o *Desenvolvimento do Sistema* onde é descrita a implementação do sistema, tanto no lado do cliente como servidor, assim como a estrutura da Base de Dados (BD).

Por fim, o capítulo 5 contém a conclusão deste projecto, onde são avaliados os resultados e propostas novas funcionalidades para trabalho futuro.

2 Estado da Arte

Ao longo deste capítulo, serão nomeados os principais *corpora* linguísticos multimédia, assim como, serão também descritos os *corpora* multimédia multilingues em termos de funcionalidades das suas respectivas interfaces.

Actualmente, existem vários *corpora* linguísticos multimédia, tais como: o EPIC (*European Parliament Interpreting Corpus*) [2], um *corpus* multilingue de discursos do Parlamento Europeu; o *Yiddish Corpus* [3], *corpus* da língua Iídiche, língua germânica das comunidades judaicas da Europa Central e Ocidental [4]; o *Childes* [5], *corpus* de discursos de crianças; o *SCOTS* (*Scottish Corpus Of Texts Speech*) [6], um *corpus* que tem como intuito o estudo do Escocês; o *TCSE* (*TED Corpus Search Engine*) [7], *corpus* bilingue de discursos das conferências TED (*Technology, Entertainment and Design*); o *FLLOC* (*French Learner Language Oral Corpora*) [8], *corpus* cujo objectivo é promover o estudo da aprendizagem do Francês como língua estrangeira; *SPLLOC* [9] (*Spanish Learner Language Oral Corpora*) [10], com o mesmo intuito do *corpus* anterior, no entanto, neste caso, a língua estudada é o Espanhol; e, por fim, o *corpus VEIGA* [11], um *corpus* bilingue de legendas em Inglês e Galego.

Uma vez que o *corpus* que dá tema a esta dissertação é bilingue, apenas os *corpora* multilingues serão descritos ao longo deste capítulo. Nestes parâmetros encaixam-se três *corpora* dos referidos anteriormente: o *corpus VEIGA* e TCSE, bilingues, e o EPIC, trilingue.

2.1 TCSE

Este projecto utiliza os discursos proferidos nas conferências TED (*Technology, Entertainment and Design*) como *corpus*. O TCSE (*TED Corpus Search Engine*) [7] disponibiliza estes mesmos discursos em formato vídeo com as respectivas transcrições e traduções, albergando cerca de dois mil discursos na sua Base de Dados, em 28 línguas diferentes, incluindo o português.

2.1. TCSE

A aplicação *web* do TCSE oferece dois tipos de pesquisa, a pesquisa simples e a avançada. No caso de se realizar uma pesquisa simples, é possível que esta seja bilingue se a opção "*translation*"for seleccionada como "*Search Target*". Na pesquisa simples, é possível ainda seleccionar opções para incluir apenas discursos em Inglês, utilizar segmentos expandidos, o que permite obter um melhor contexto no qual a palavra (ou expressão) pesquisada se insere, começar a reprodução do vídeo 10 segundos mais cedo, escolher a velocidade de reprodução, a qualidade e a respectiva fonte (TED ou *Youtube*) do mesmo, e ainda que este pare automaticamente.

Na pesquisa avançada, o TCSE aceita operadores de pesquisa que permitem que esta deixe de estar limitada a uma busca pela exacta palavra (ou expressão) inseridas, como acontece na pesquisa simples [12]. Os operadores existentes permitem realizar uma pesquisa por lemas [13], ou seja, pela forma canónica da palavra, por partes do discurso, por *surface text forms* ou ainda, pesquisar utilizando uma combinação dos operadores.

Após a realização de uma pesquisa, são listados os segmentos de texto que incluem a palavra ou expressão inserida (figura 2.1). Para cada segmento apresentado, é possível consultar metadados do discurso e a sua sinopse, consultar apenas a transcrição do discurso, ou visualizar o discurso com as respectivas legendas, no caso de uma pesquisa bilingue estas aparecem lado a lado (figura 2.2). Tanto a opção de reproduzir o discurso como a de apenas consultar o mesmo em texto, abrem uma nova janela de navegação no *browser*.

Figura 2.1: Página de resultados do TCSE

233	10:15	**And there's a beautiful experiment in neuroscience to illustrate this.**	E há uma bela experiência na neurociência que ilustra isso.
234	10:19	**And unlike most neuroscience experiments,**	E, ao contrário da maioria das experiências neurocientíficas,
235	10:21	**this is one you can do at home.**	
236	10:23	**All you need is one of these.**	esta, vocês podem fazer em casa.
237	10:25	**(Laughter)**	Tudo o que precisam é de uma destas.
238	10:26	**And a couple of paintbrushes.**	(Risos) E alguns pincéis.
239	10:29	**In the rubber hand illusion,**	Na ilusão da mão de borracha,

Figura 2.2: Página de visualização de um discurso no TCSE

2.2 Veiga

Desenvolvido ao abrigo do CLUVI (*Corpus* Linguístico da Universidade de Vigo), um conjunto de *corpora* paralelos especializados na língua galega contemporânea [14] produzido pelo SLI (Seminário de Linguística Informática) da Universidade de Vigo, o *corpus Veiga* representa um *corpus* multimédia de lengendagem em inglês e em galego.

O *corpus Veiga* [11], actualmente, constituído por 24 filmes legendados em inglês e em galego, permite apenas pesquisa bilingue, não havendo a opção de executar uma pesquisa monolingue. Para realizar uma pesquisa, basta inserir uma palavra em inglês ou galego na barra de pesquisa correspondente a cada língua (figura 2.3), depois, os resultados são apresentados numa lista (figura 2.4). Para cada item da lista de resultados, é possível consultar o contexto do segmento apresentado ao clicar no botão com a seta, ou então, visualizar o excerto do filme onde é referido o segmento de texto, em ambos os casos, é aberta uma nova janela.

No caso da visualização de discursos (figura 2.5), a página está dividida em duas secções, uma em que contém o clipe de vídeo com as legendas em inglês e outra secção com o mesmo clipe mas com as legendas em galego. Por uma questão de contexto, é possível seleccionar o clipe de vídeo anterior e posterior ao clipe onde foi referido o excerto de texto seleccionado anteriormente. Para ser possível visualizar os vídeos, é necessária a instalação do *flash player*.

Ao realizar a pesquisa, para além de pesquisar por uma palavra ou expressão, é também

possível inserir expressões regulares do tipo PCRE (*Perl Compatible Regular Expressions*), o que permite uma pesquisa mais aprofundada.

Figura 2.3: Página de pesquisa do *corpus Veiga*

1-AXU (53)	She's comin' up to help with ¶ the kids.	Vai vir **axudar** cos nenos.		
2-AXU (527)	I don't know how to help you.	-Non sei cómo **axudar**lle.		
3-AXU (567)	I need some help tonight.	Hoxe necesito **axuda**.		
4-AXU (699)	-How can you help me?	-Como me vas **axudar**?		
5-AXU (768)	I can't help you...	¶ Non lle podo **axudar**.		

Figura 2.4: Lista de resultados do *corpus Veiga* após a pesquisa da palavra galega "*axuda*"

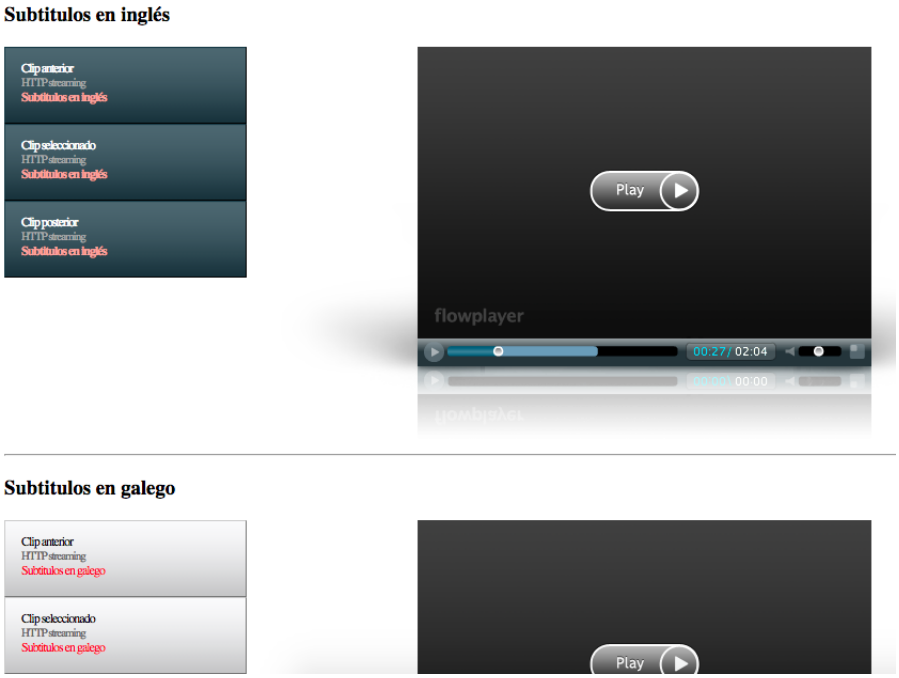


Figura 2.5: Visualização de discursos do *corpus Veiga*

2.3 EPIC

O *EPIC* [15] é um *corpus* trilingue (Italiano, Inglês e Espanhol) de discursos do Parlamento Europeu e das suas respectivas transcrições [2]. Este é constituído por três *subcorpora* de textos originais (Original-Italiano, Original-Inglês, Original-Espanhol) e seis *subcorpora* de textos interpretados (Espanhol-Italiano, Espanhol-Inglês, Italiano-Inglês, Italiano-Espanhol, Inglês-Italiano, Inglês-Espanhol).

O primeiro tipo de *subcorpus* referido (*source texts*), corresponde a vários *corpora* monolingues que permitem a pesquisa de *queries* nos discursos das três línguas disponíveis de fonte original. O segundo tipo de *subcorpus* referido (*target texts*) é contituído por *corpora* onde é possível realizar comparações entre discursos de fonte original com a sua respectiva interpretação noutra língua. Uma vez que o EPIC é trilingue, é possível comparar o mesmo discurso original a duas interpretações em línguas diferentes.

Para realizar uma pesquisa, tanto nos *subcorpora* "*source texts*"como nos "*target texts*"é necessário realizar *login*, no entanto, como os novos registos não estão activos, não é possível fazer uma descrição ao nível de aplicação deste *corpus*. Todavia, é referido no *website* do *EPIC*

[2] que este apresenta vários parâmetros de pesquisa, tais como: a velocidade do discurso, a categoria do tópico do discurso, características do orador, género e ainda, país de origem. Este também permite que os utilizadores escolham o que exibir nos resultados: palavras, lemas, forma como a palavra foi pronunciada e etiquetas de *part-of-speech*.

2.4 Análise Comparativa

Dos três *corpora* analisados neste capítulo, podemos verificar que nenhum tem as seguintes características:

- Interface simples e intuitiva;
- O português presente tanto como língua original como interpretada, sendo que dos *corpora* mencionados apenas o TCSE contém discursos em português, no entanto, apenas como língua de interpretação. Ou seja, nenhum é um *corpus* portugês bidireccional.

Tendo em conta o referido anteriormente, o *interPE* vem propôr um *corpus* de interpretação multimédia bidireccional (português-inglês) do Parlamento Europeu, com uma interface simples.

2.5 Tecnologias *web*

As tecnologias *web* actuais e relevantes no desenvolvimento do *interPE* são as seguintes:

- HTML5: O HTML5 [16] corresponde à versão mais recente do HTML (*Hypertext Markup Language*) [17], uma linguagem de marcação essencial para o desenvolvimento de páginas *web*. Este traz novas características à linguagem de marcação, como por exemplo, o facto de oferecer suporte para áudio e vídeo.

  Apenas os ficheiros de áudio nos formatos *mp3*, *wav* (com o *codec PCM*) e *ogg* (com os *codecs FLAC* ou *Opus*) são suportados [17], sendo que apenas o *mp3* é suportado pelos browsers principais [18]. No caso dos ficheiros de vídeo, são suportados os formatos *WebM* (com os *codecs VP8* ou *VP9* de vídeo e os *codecs Vorbis* ou *Opus* de áudio), *ogg* (com o *codec* de vídeo *Theora* e o *codec* de áudio Vorbis), *mp4* (com o *codec H.264* de vídeo e o *codec AAC* de áudio), no entanto, apenas o formato *mp4* é suportado pelos *browsers* principais [19].

- CSS: O CSS (*Cascading Style Sheets*) [20] é uma linguagem que trata de descrever a forma como os elementos, neste caso, HTML devem ser apresentados. No caso das aplicações *web*, o CSS é o responsável pela definição do aspecto da página, ou seja, é através deste que podemos alterar o visual da aplicação pela possibilidade de definição de vários estilos, entre eles: o tamanho ou estilo da fonte, cores utilizadas ou até dimensão e posição dos elementos HTML definidos.

- JavaScript: O *JavaScript* [21] é uma linguagem de programção interpretada e orientada a objectos. Incialmente foi criada para que os seus *scripts* pudessem ser executados do lado do cliente, em *browsers*, no entanto, actualmente já é utilizada também no lado do servidor em ambientes como *Node.js* [22] ou *Commonjs* [23], o que simplifica o processo de desenvolvimento e manutenção. Actualmente, existem várias *front-end frameworks* de *JavaScript* que permitem o desenvolvimento de aplicações *web* complexas e sofisticadas em termos de dinamismo da interface, facilitando todo o processo de construção das mesmas. A existência de tais *frameworks* facilitou o desenvolvimento de *Single Page Applications*, tornando este tipo de aplicações mais comum no mundo da *web*.

- NoSQL: *NoSQL* [24] corresponde a um modelo de Base de Dados não relaccional, portanto, a base de dados não seguirá o modelo tradicional relacional, tal como referido em "*What is a non relational database*"[25]. Ou seja, ao contrário das BD relaccionaois, os dados deixam de estar organizados em tabelas que se relacionam entre si e passam a estar organizados em colecções, sem relação. As coleccções também são mais flexíveis que uma tabela, uma vez que os elementos da mesma não têm obrigatoriamente de ter os mesmos campos. Há várias categorias de BD não relacionais consoante o método para o armazenamento de dados, tais como, documento, gráfico, coluna, entre outros.

- JSON: JSON (*JavaScript Object Notation*) [26] é um formato para o intercâmbio de dados, com a característica de fácil leitura ou escrita do mesmo por parte de um humano. Apesar de ser nativo no *JavaScript*, pode ser utilizado independentemente da linguagem de programação [27]. No caso das aplicações *web*, este poderá ser utilizado para o formato escolhido para a troca de dados entre o cliente e o servidor.

- HTTP: O *Hypertext Transfer Protocol* (HTTP) [28], criado em 1990 com o objectivo de

ser utilizado na troca de dados entre um *web browser* e um *web server*, é um protocolo da camada de aplicação cujo modelo é "cliente-servidor". Neste protocolo, o cliente envia pedidos, *HTTP Requests*, para o servidor, que após executar as tarefas necessárias, responde com um *HTTP response* contendo os dados requisitados.

3 Análise do Sistema

3.1 Requisitos do Sistema

O objectivo do desenvolvimento de uma aplicação *web* que serve de suporte ao *corpus* do Parlamento Europeu é a integração de várias funcionalidades de pesquisa e manutenção do mesmo, tornando a consulta deste mais simples e rápida, optimizando-a.

Os requisitos deste sistema são os seguintes:

- Interface intuitiva — a aplicação web deve ter uma interface de utilizador simples e intuitiva, de modo a tornar a utilização da aplicação acessível a qualquer pessoa, tendo esta ou não conhecimentos na área da Linguística;

- Pesquisa bilingue — quando um utilizador faz uma pesquisa numa dada língua, os resultados apresentados, para além de serem mostrados para essa língua, também revelam o discurso correspondente interpretado, ou original, dependendo da situação;

- Modo de pesquisa avançada — o utilizador deve ter a opção de especificar os dados da sua pesquisa, sendo possível filtrar a mesma através dos metadados associados a cada discurso, entre eles o nome do orador, género, tópico do discurso, grupo político, data e ainda, *speech ID*;

- Apresentação de resultados tanto para os discursos originais como interpretados — se o utilizador realizar uma pesquisa numa determinada língua é possível escolher se este pretende visualizar os resultados obtidos quando a língua seleccionada corresponde à fonte original do discurso ou quando corresponde à fonte interpretada, bastando carregar num botão que alterna entre os dois tipos de resultados;

- Visualização do discurso — após realizar a pesquisa, o utilizador pode seleccionar um dos resultados e visualizar o discurso em formato vídeo, assim como as respectivas

transcrições do discurso original e interpretado sincronizadas com o mesmo;

- Apresentação do discurso em formato texto — após a realização da pesquisa, caso o utilizador pretenda consultar apenas a transcrição do discurso em formato texto e não a opção multimédia, esta deve ser possível;

- Consulta dos metadadados associados a cada discurso, quer a fonte seja interpretada ou original;

- Adição de novos discursos do Parlamento Europeu por parte do administrador, inserindo os ficheiros respectivos a um discurso (ficheiro com metadados, áudio, vídeo, *snippet* do vídeo e transcrições), de modo a manter o sistema actualizado.

Inicialmente, para além da pesquisa bilingue foi estabelecido como requisito do sistema a pesquisa monolingue. No caso deste modo de pesquisa, nos resultados apenas constaria a transcrição do discurso com a língua correspondente à seleccionada pelo utilizador, sem que fosse possível visualizar a respectiva fonte original ou interpretada do mesmo, consoante a situação. Após alguma reflexão, foi concluído que a pesquisa monolingue não apresenta vantagens sobre a pesquisa bilingue, uma vez que, em termos linguísticos, a possibilidade de comparação de duas línguas torna a aplicação do *corpus* mais interessante. Como tal, a pesquisa monolingue não consta nos requisitos do sistema.

3.2 Casos de Uso

As funcionalidades da aplicação nas diferentes interacções com os dois tipos de utilizadores, comum e administrador, são formalizadas em vários casos de uso, que representam as diversas situações enunciadas na secção "Requisitos do Sistema"(secção 3.1).

A figura 3.1 apresenta as possiblidades de acção na página principal. No caso do utilizador comum, estas são: execução de uma pesquisa simples, execução de pesquisa avançada e consulta de informações sobre a aplicação; no caso do administrador, para além das acções que um utilizador comum pode realizar, este pode, ainda, adicionar novos discursos.

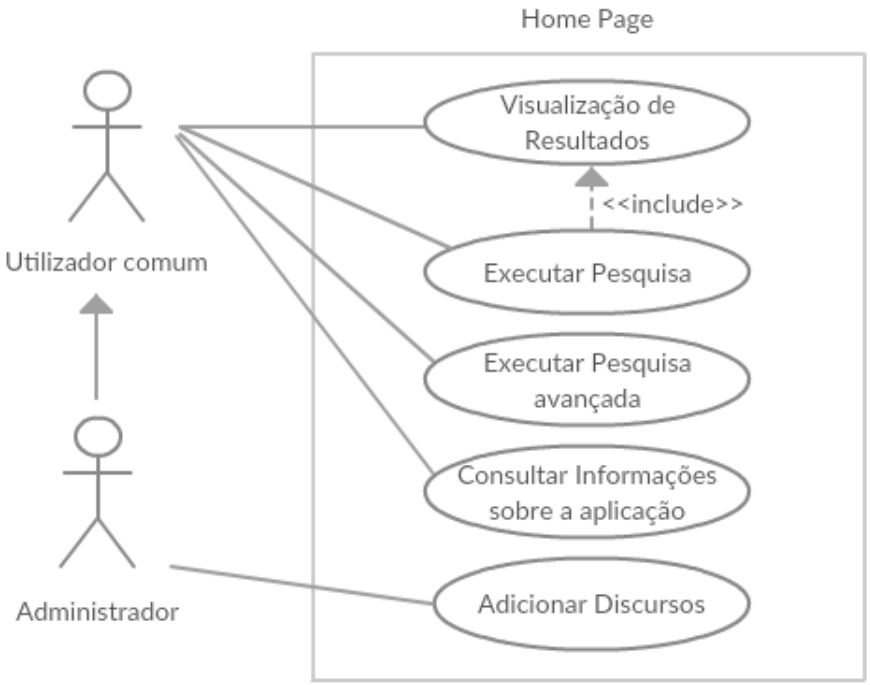


Figura 3.1: Diagrama Casos de Uso Página principal

Após a execução de uma pesquisa, simples ou avançada, são apresentados os resultados obtidos, onde é possível a visualização multimédia do discurso, a consulta em texto das transcrições, a consulta dos metadados associados a cada discurso e a alteração da pesquisa. Estas acções encontram-se representadas na figura 3.2.

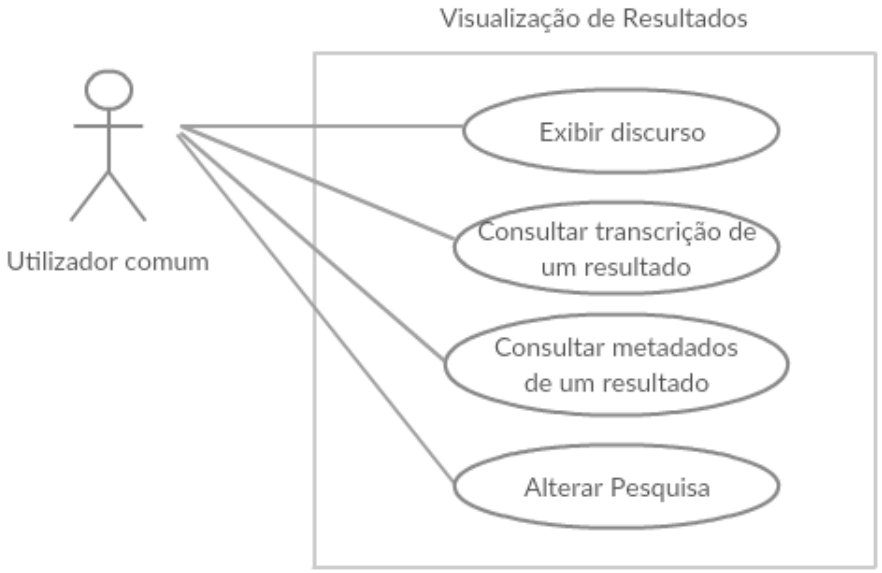


Figura 3.2: Diagrama Casos de Uso Visualização de Resultados

Para garantir que apenas um utilizador administrador tem acesso à função de adicionar novos discursos ao *corpus*, antes do acesso a esta funcionalidade é apresentado um formulário de *login*, tal como representado na figura 3.3.

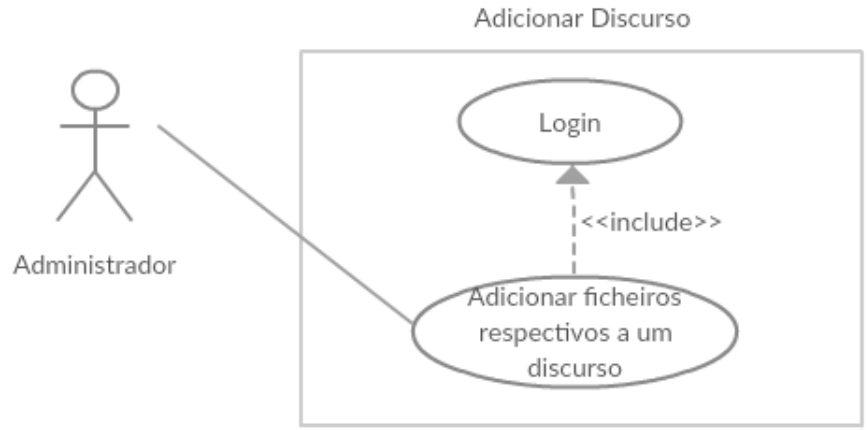


Figura 3.3: Diagrama Casos de Uso Adição Novo Discurso

3.3 Arquitectura do Sistema

A aplicação *web* é uma SPA (*Single Page Application*). A arquitectura deste sistema é "cliente-servidor", o modelo mais popular utilizado actualmente na *world wide web* [29], uma

vez que torna possível a execução simultânea de vários clientes sem comprometer o bom funcionamento do sistema.

O sistema *interPE* é constituído pelo cliente *web*, o servidor e a Base de Dados. O cliente comunica com o servidor através de HTTP *requests*, por sua vez, o servidor responde utilizando HTTP *responses*. Já a comunicação entre o servidor e a BD é realizada enviando *queries* do servidor para a Base de Dados.

O servidor é o intercomunicador entre o cliente e a BD, uma vez que uma das funções principais do *interPE* é a pesquisa de palavras (ou expressões) nas transcrições armazenadas na mesma, portanto, a maioria dos pedidos realizados pelo cliente requerem a obtenção de informação da Base de Dados. Para além de comunicar com a BD, o servidor serve o cliente na reprodução de áudio e vídeo dos discursos.

A arquitectura do *interPE* está representada na figura 3.4.

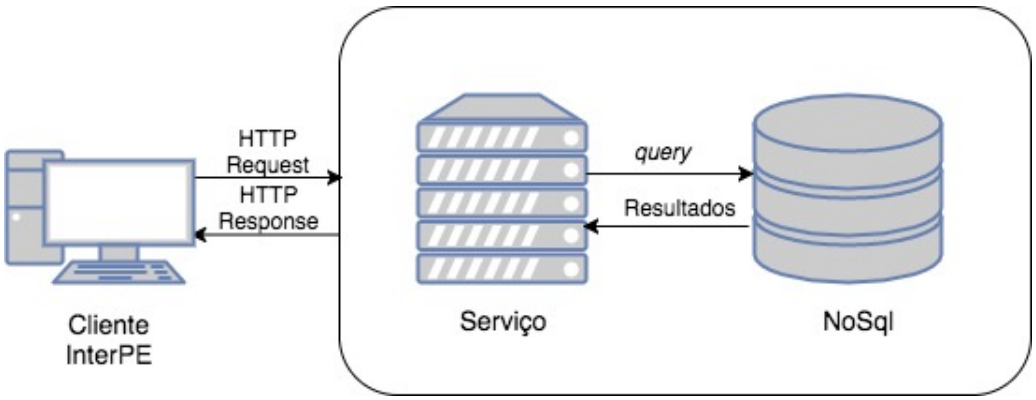


Figura 3.4: Arquitectura do *interPE*

3.4 Concepção da Interface

3.4.1 Navegação

A interface do *interPE* será constituída por 4 páginas: a *home page*, a página para adicionar novos discursos, a página de contactos e a página *about*. O diagrama da figura 3.5 representa o esquema de navegação da aplicação. Uma vez que, em todas as páginas estará presente uma barra de navegação, a qualquer momento é possível alternar entre as páginas da aplicação.

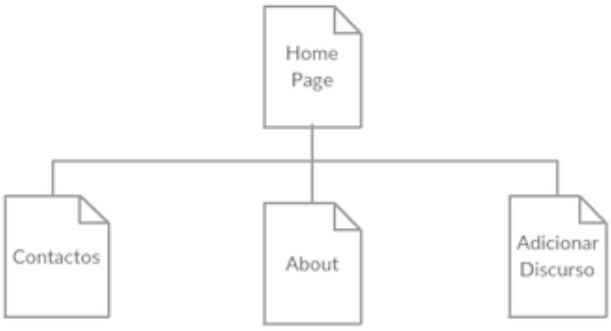


Figura 3.5: Diagrama de Navegação do *interPE*

Apesar da existência de apenas quatro páginas, uma vez que esta é uma SPA, a *home page* alterará o seu estado consoante as acções do utilizador, sem a necessidade de navegação para uma nova página. O diagrama da figura 3.6 mostra a sequência dos diferentes estados da *home page*.

Figura 3.6: Diagrama dos diferentes estados da *home page*

O estado de pesquisa corresponde ao estado apresentado incialmente pela *home page*.

3.4.2 Esboços da Interface

A *home page* da aplicação será constituída por uma barra de navegação, o nome da aplicação e uma barra de pesquisa. Na figura 3.7 está representado um esboço da mesma.

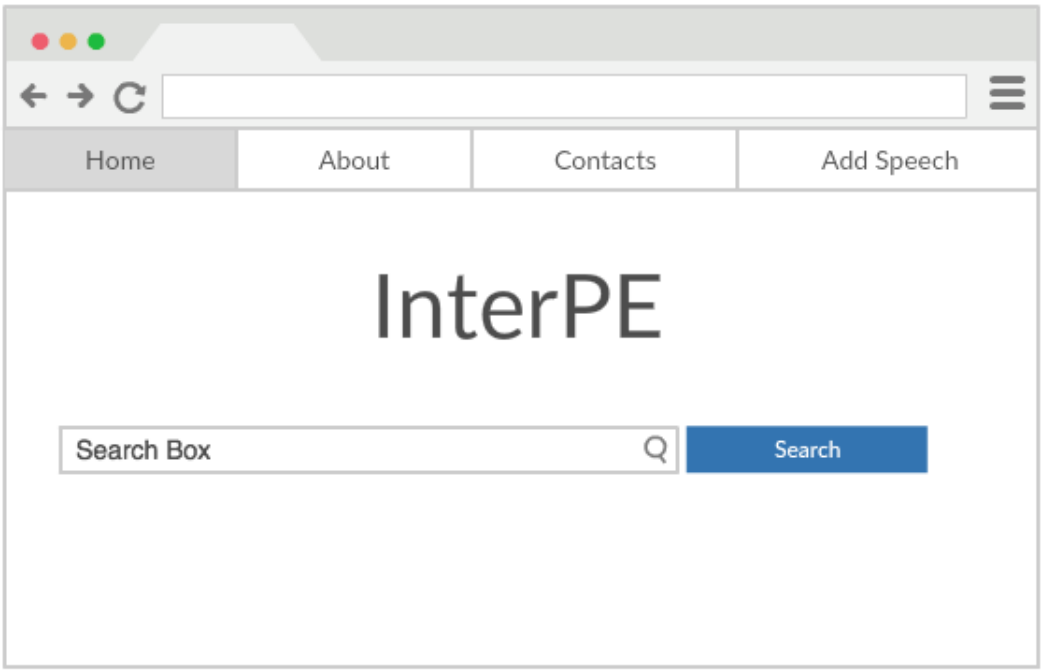


Figura 3.7: Esboço *Home Page* do *interPE*

Tal como mencionado na secção anterior, a *home page*, após a pesquisa ser realizada, altera o seu estado para mostrar os resultados, conforme ilustrado na figura 3.8. A visualização de resultados será constituída por uma tabela para listar os resultados obtidos, constituída por uma coluna com a opção de exibir o discurso e duas colunas com um excerto do discurso original e interpretado. Será ainda possível consultar os metadados do discurso, assim como as transcrições em texto.

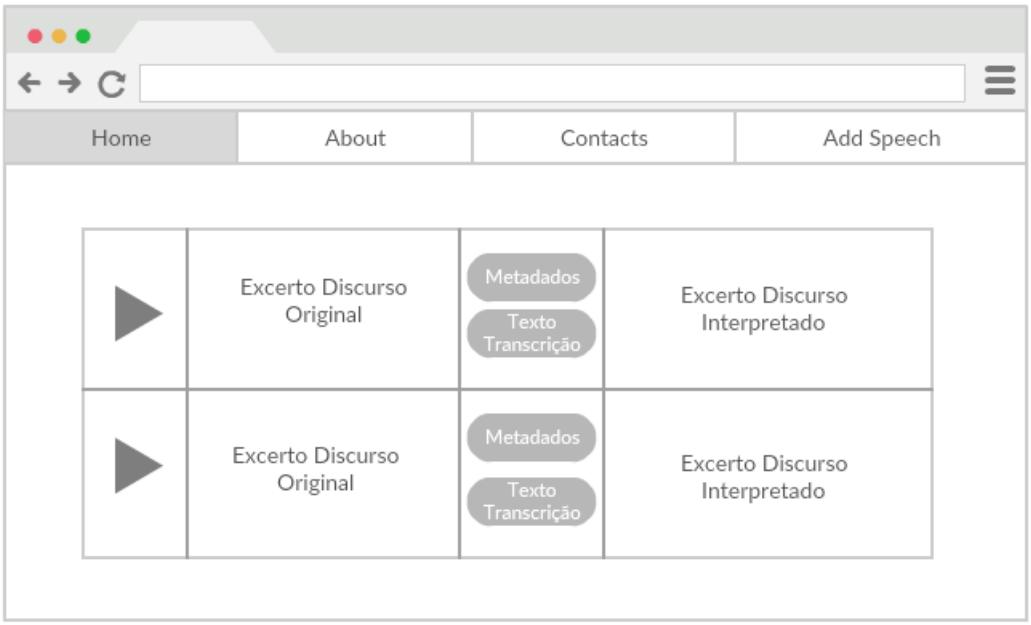


Figura 3.8: Esboço visualização de resultados do *interPE*

A partir da *home page* apresentar a lista de resultados, é possível exibir um discurso. Neste caso, a página será consituída por um vídeo, duas faixas de áudio e, ainda, as transcrições original e interpretada do discurso, tal como se pode observar na figura 3.9.

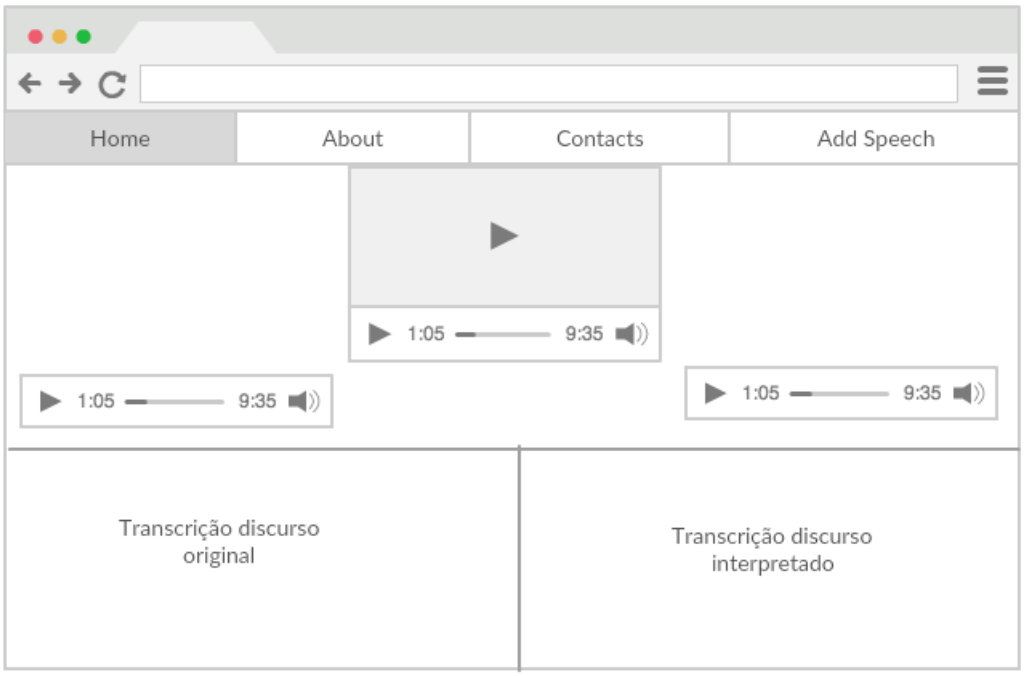


Figura 3.9: Esboço visualização de um discurso do *interPE*

Tanto a página de contactos como a *about* seguem a mesma estrutura: barra de navegação, o título e um texto com as devidas informações. O esboço das páginas está representado na figura 3.10.

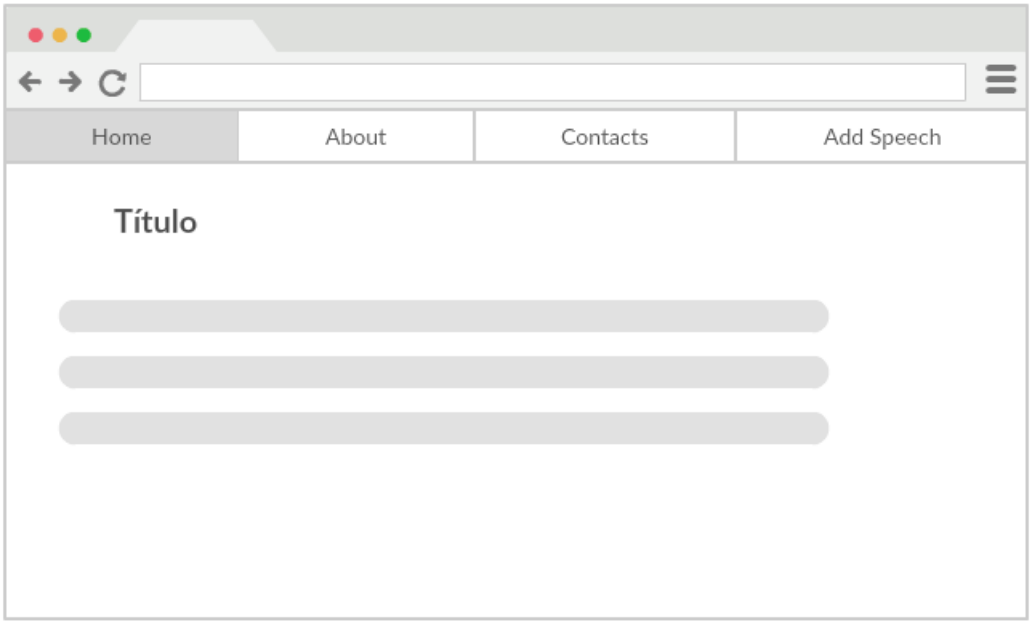


Figura 3.10: Esboço página de Contactos e *About* do *interPE*

Por fim, para adicionar novos discursos, a página será constituída por um formulário com vários campos e, tal como nos casos anteriores, uma barra de navegação, tal como é possível observar na figura 3.11.

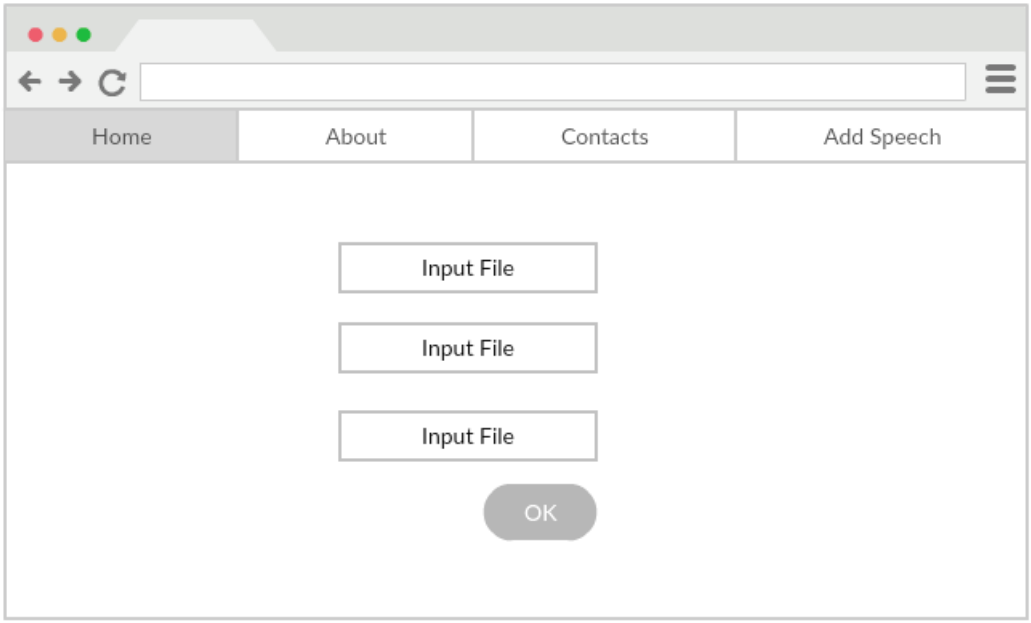


Figura 3.11: Esboço página de adição de discursos do *interPE*

3.5 Tecnologias a utilizar

3.5.1 Escolha *Front-end Framework*

Actualmente, existem variadíssimas *front-end frameworks* de JavaScript, entre elas: Angular.js, React.js, Vue.js, Meteor, Ember.js, etc. Todas elas têm o objectivo comum de facilitar a construção das interfaces de utilizador, agilizando todo o processo de desenvolvimento das mesmas. Após um período de pesquisa, a *framework* escolhida para este projecto foi Vue.js [30], na versão *2.2.1*.

Vue.js é uma *front-end framework* cuja arquitectura se baseia parcialmente no modelo MVVM (*Model-view-viewmodel*) [31], sendo utilizada para a construção de componentes reactivos. Nesta *framework*, cada instância do *Vue* é um *ViewModel*, a DOM gerida pelas instâncias do *Vue* correspondem ao *View* e os *Models* são simples objectos de JavaScript, ou *data objects* [31].

Para cada componente, o Vue.js utiliza um ficheiro de extensão *.vue* cuja estrutura se divide em três secções:

- *template*, sendo que é possível a utilização de qualquer linguagem de *template* como

HTML, Jade, entre outras;

- *script*, onde é definido o comportamento do componente;
- *style*, que trata de estilizar o *template*.

Na figura 3.12, pode-se observar um ficheiro exemplificativo do referido.

```
<template>
    <div>
        <h3>Tell us what you think:</h3>
        <textarea id="opinion_area" rows="4" cols="50"> </textarea>
        <button type="button" @click="send">Submit opinion</button>
    </div>
</template>

<script>
    export default{
        methods: {
            send(){
                //método que envia a submissão do utilizador para o servidor
            }
        }
    }
</script>

<style>
    #opinion_area{
        margin-bottom: 10px;
        border-color: black;
    }
</style>
```

Figura 3.12: Ficheiro simples vue.js

A escolha desta *framework* JavaScript em detrimento das outras baseia-se nos seguintes factos:

- A curva de aprendizagem é curta em comparação com outras *frameworks* do género, Vue.js é mais rápido de aprender;
- A documentação é muito completa e explícita;

- Apesar da comunidade ser menor que de outras ferramentas similares, é muito activa e colaborativa;
- Apresenta uma boa *performance* em comparação com outras *frameworks* já existentes [32];
- Por fim, exibe um grande crescimento, tendo já sido adoptado como a *framework* JavaScript oficial do Laravel, uma das mais populares *frameworks* de PHP [33];

3.5.2 Escolha da Base de Dados

Uma Base de Dados (BD) é uma colecção de dados organizada que permite o acesso, gestão ou actualização fácil de dados [34]. Neste trabalho, foram considerados dois dos principais modelos arquitecturais: o modelo relacional e o não relacional.

Uma Base de Dados relacional corresponde a uma estrutura de dados organizada em tabelas, constituídas por colunas e linhas [35]. As linhas representam instâncias da entidade e as colunas representam valores associados a essas instâncias. As tabelas relacionam-se entre si através de chaves denominadas por estrangeiras, sendo que cada linha é constituída por uma instância única. A linguagem utilizada para realizar as *queries* à BD é o SQL (*Structured Query Language*).

Uma BD não relacional, tal como referido em "Estado da Arte", não segue o modelo tradicional relacional. Portanto, ao contrário do modelo tradicional SQL (*Structured Query Language*), em que os dados estão organizados em tabelas relaccionadas entre si, neste caso, os dados são organizados em colecções, não havendo relações entre as mesmas.

Por exemplo, para guardar uma lista de compras numa BD relacional seria criada uma entidade "lista", que possuíria o seu *id* único e a data em que esta foi efectuada. A segunda entidade possuíria também um *id* único, um atributo "nome", outro "quantidade"e uma chave estrangeira para a "lista". Após criada esta estrutura, todos os objectos devem obedecer-lhe. Um exemplo da arquitectura da BD está representado na figura 3.13. No caso de uma BD não-relaccional, se, por exemplo, utilizássemos o *MongoDB*, bastaria a criação de um documento "lista", com o seu *id* único, data e um documento produto, que serão armazenados numa colecção de listas, tal como se mostra na figura 3.14.

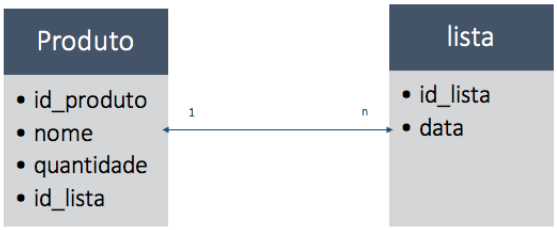


Figura 3.13: Estrutura da lista de compras utilizando SQL

```
{
    _id: id_lista
    "data": new Date(2017,3,25),
    "produto": {"nome":"manteiga", "quantidade": 1}
}
```

Figura 3.14: Estrutura da lista de compras utilizando NoSQL

Tal como se pode observar na figura 3.14, o documento "produto", uma vez que está embebido no documento "lista", não precisa de um *id* único.

As vantagens apresentadas pelas bases de dados relacionais [36] baseiam-se no facto destas apresentarem as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade). A atomicidade garante que se um elemento de uma operação falhar, todos os dados são repostos de forma a que não haja alterações na BD; a consistência certifica que uma transação ou produz um novo estado de informação ou, em caso de falha, repõe todas as alterações para o seu estado anterior; o isolamento assegura que uma transação, que ainda esteja em processo, se mantém isolada das restantes; por último, a durabilidade garante que alterações realizadas por uma transação são guardadas mesmo que a BD falhe por algum motivo.

Já as vantagens características de uma BD NoSQL [37] são a simplicidade do seu design, o facto de serem escaláveis horizontalmente, ou seja, facilmente distribuem os dados armazenados por vários servidores (sistema distribuído); são mais rápidas que as relacionais, uma vez que não é necessário fazer *join* às várias tabelas; é possível guardar qualquer tipo de dados sem tornar a BD lenta; e por último, é possível alterar a base de dados sem a pôr *offline*.

Num mundo como o da *web*, onde se preza a rapidez de acesso à informação e a escalabilidade, a integridade das bases de dados relacionais é posta de lado em detrimento da rapidez e possibilidade de servir um número de pedidos crescente que as BD não relacionais proporcionam, uma vez que estas não são horizontalmente escaláveis. Apesar de não ser impossível escalar uma BD relacional horizontalmente, a sua implementação é muito complexa devido ao facto de manter a integridade dos dados, logo, a escolha de um modelo arquitectural, neste caso, parte sempre para as BD NoSQL.

Portanto, tendo em conta os factos referidos ao longo deste capítulo, o modelo escolhido foi o não relacional sendo a plataforma adoptada o *MongoDB* [38], a mais comum no universo NoSQL. Esta guarda a informação em ficheiros do tipo BSON (Binary JSON) [39], o que permite adicionar novos tipos de dados para além dos nativos do JSON [40], tais como o *ObjectID* ou a data. No *MongoDB*, os dados são organizados em colecções, o que equivale a uma tabela SQl, e documentos, equivalentes à linha de uma tabela. Uma vez que o *MongoDB* oferece uma estrutura dinâmica, os documentos podem ter campos diferentes, não havendo a necessidade de definir uma estrutura inicial.

4 Desenvolvimento do projecto

4.1 Estrutura da Base de Dados

A BD do *interPE* é constituída unicamente por uma colecção "vídeo", sendo esta responsável por armazenar todos os dados relaccionados com os discursos pertencentes a este *corpus*. Cada documento associado a um discurso original é constituído pelos seguintes campos:

- *Metadata*: um objecto que guarda os metadados de cada discurso. Os campos do mesmo são: *SpeechID*, o identificador de cada discurso; a *source*, isto é, se a fonte do discurso é original ou interpretada; *date*, ou seja, a data do mesmo; *language*, a lingua em que o discurso foi proferido; *topic*, o tópico de discussão; *speaker*, o nome do deputado que discursa; *politicalGroup*, o partido político a que o deputado está associado; *gender*, o género do deputado;
- *Video*: o nome do ficheiro de vídeo associado ao discurso;
- *Audio*: o nome do ficheiro áudio associado ao discurso;
- *Snippet*: o nome do ficheiro com a imagem identificadora do vídeo referido;
- *Lgnd_name*: o nome do ficheiro que contém a transcrição do discurso;
- *Lgnd*: o conteúdo do ficheiro acima mencionado.

Todos os ficheiros referidos são armazenados numa pasta denominada por *media*, no servidor.

No caso do discurso não ser original mas sim interpretado, os campos *Video* e *Snippet* não farão parte da constituição do documento, pois, ao visualizar um discurso é apresentado apenas um vídeo. Ora, uma vez que entre o discurso original e o correspondente interpretado a imagem do vídeo é igual, sendo a faixa de áudio distinta, não é necessário guardar duas vezes

a mesma informação ainda que em documentos diferentes. Assim, evita-se a redundância de dados na BD, mantendo estes dados apenas nos documentos de discursos originais.

Quanto ao *Snippet*, uma vez que este contém uma imagem representativa do vídeo em questão, só fará sentido que este campo exista quando o campo *Video* estiver presente. Para além dos dois campos mencionados, os campos *speaker* e *politicalGroup* do objecto *Metadata* também não fazem parte dos documentos que contêm discursos interpretados.

A razão pela qual o conteúdo do ficheiro com as transcrições dos discursos fica directamente armazenado num campo da BD é para evitar numerosas operações de *read/write* do ficheiro uma vez que estas são operações pesadas, assim, por exemplo, de cada vez que é realizada uma pesquisa pelo utilizador, é evitado o processo de *read*, bastando o acesso à string guardada no campo *lgnd*. Esta solução é viável uma vez que o conteúdo das transcrições ocupa um espaço reduzido na Base de Dados.

Tal como já foi referido em capítulos anteriores, o *interPE* é um *corpus* multimédia cujo único modo de pesquisa é o bilingue, portanto, haverá sempre um discurso original associado a um interpretado aquando da visualização dos resultados. Esta associação é possível através do *SpeechID* de cada discurso; este *ID* foi criado pela Ana Correia, que incialmente codificou este *corpus*, doutoranda do Instituto de Letras e Ciências Humanas da Universidade do Minho para estudar a anáfora na interpretação simultânea. O *SpeechID* é constituído por 7 caracteres, em que os primeiros dois correspondem ao número atribuído ao deputado na lista global de eurodeputados, os dois seguintes são as iniciais do nome do deputado, o quinto carácter corresponde ao número do discurso feito pelo deputado e, por fim, os últimos dois correspondem à língua da transcrição. Por exemplo, para o caso do *SpeechID* "11LA6PT", o "11"indica o número do deputado na lista global de eurodeputados portugueses, o "LA"corresponde às iniciais do deputado, neste caso, Luís Alves; o "6"indica o número do discurso na lista global de discursos feitos pelo Luís Alves; por fim, o "PT"indica que o discurso foi transcrito em português. Deste modo, é possível associar um discurso original a um interpretado, uma vez que o *SpeechID* destes discursos difere apenas na língua, indicada pelos dois últimos caracteres.

4.2 Cliente

Neste capítulo, será abordada a implementação das funcionalidades e respectivas interfaces do cliente.

4.2.1 Componentes

O objectivo do desenvolvimento da aplicação *web interPE*, passa pela construção de uma *Single Page Application* (SPA). Esta aplicação consiste apenas numa página *web*, ao contrário dos *websites* mais comuns onde consoante as acções do utilizador novas páginas vão sendo carregadas. Neste caso, a página só é carregada uma vez, no início.

O objectivo é fornecer ao utilizador uma experiência similar a uma aplicação *mobile* ou *desktop*, tornando a página mais dinâmica [41].

Tal como referido em *Escolha Front-end Framework*, a *framework* escolhida para o desenvolvimento desta SPA foi o *Vue.js*. De acordo com a documentação do mesmo, os componentes são uma das suas características mais poderosas, uma vez que estes podem extender elementos básicos HTML, de forma a encapsular código reutilizável [42]. A concretização do comportamento de uma SPA advém da construção de vários componentes que passam a compor a interface.

Cada acção do utilizador sobre um componente despoleta um evento que resulta na execução de algo. No Vue.js, os componentes podem comunicar entre si através de duas formas:

- Eventos: um componente filho, ou sub-componente, envia um evento para o componente pai a informar de alterações que tenha sofrido;
- Props: através deste método, o componente pai pode passar dados de qualquer tipo para o componente filho.

De modo a cumprir os requisitos da aplicação, foram criados os seguintes componentes:

- *Search.vue* (figura 4.1): contém os elementos de pesquisa no *corpus*, ou seja, a barra de pesquisa, o botão com a opção de pesquisa avançada e o *modal* com as opções da pesquisa avançada;
- *Results.vue* (figura 4.2): alberga as tabelas com os resultados da pesquisa, as *tabs* para trocar entre a pesquisa nos discursos originais ou interpretados e o *modal* de visualização dos metadados dos respectivos discursos;
- *VideoPlayer.vue* (figura 4.3): engloba o *player* de vídeo e áudio, e ainda, as legendas para o discurso interpretado e original;

- *Text.vue* (figura 4.4): constituído apenas por dois campos de texto onde é possível ver o texto das transcrições originais e interpretadas de cada discurso;
- *AddSpeech.vue* (figura 4.5): contém os campos necessários para a adição de novos discursos ao *interPE*;
- *Home.vue* (figura 4.6): alberga todos os componentes referidos anteriormente, ou seja, é o componente pai de todos os descritos e, para além disso, também contém o nome e mensagem de boas vindas do *website*;
- *Nav.vue* (figura 4.7): contém a barra de navegação que permite alternar entre as "páginas";
- *Contacts.vue* e *About.vue*: inclui informações adicionais sobre a aplicação;
- *App.vue* (figura 4.8): componente pai da aplicação, todos os componentes criados são seus sub-componentes.

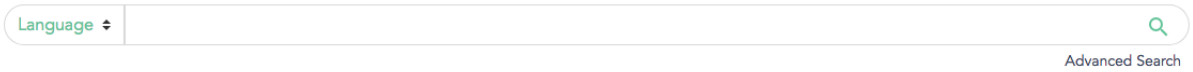


Figura 4.1: Componente *Search.vue*

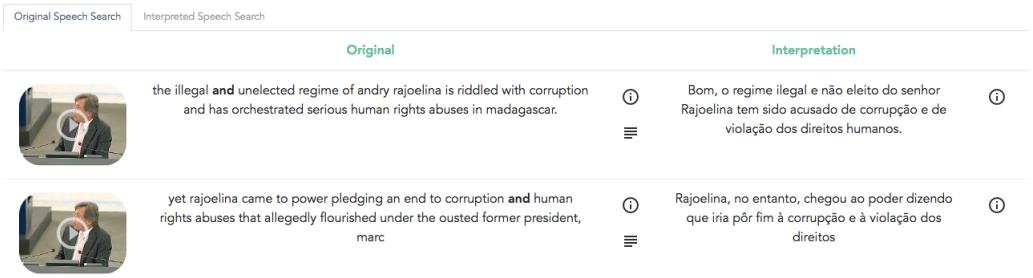


Figura 4.2: Componente *Results.vue*

4.2. Cliente

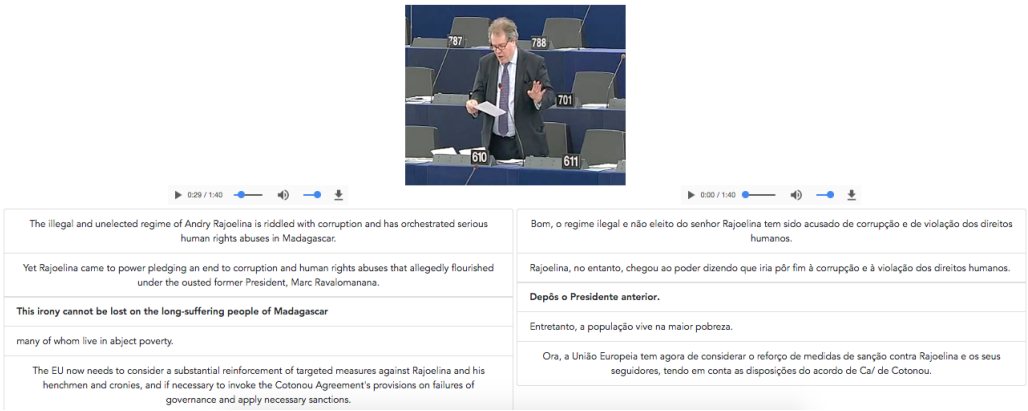


Figura 4.3: Componente *VideoPlayer.vue*

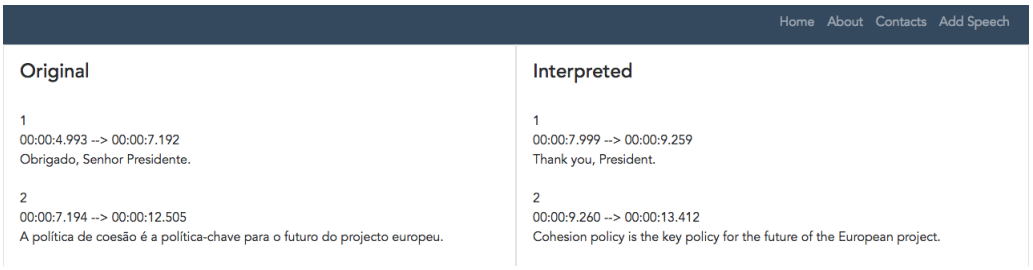


Figura 4.4: Componente *Text.vue*

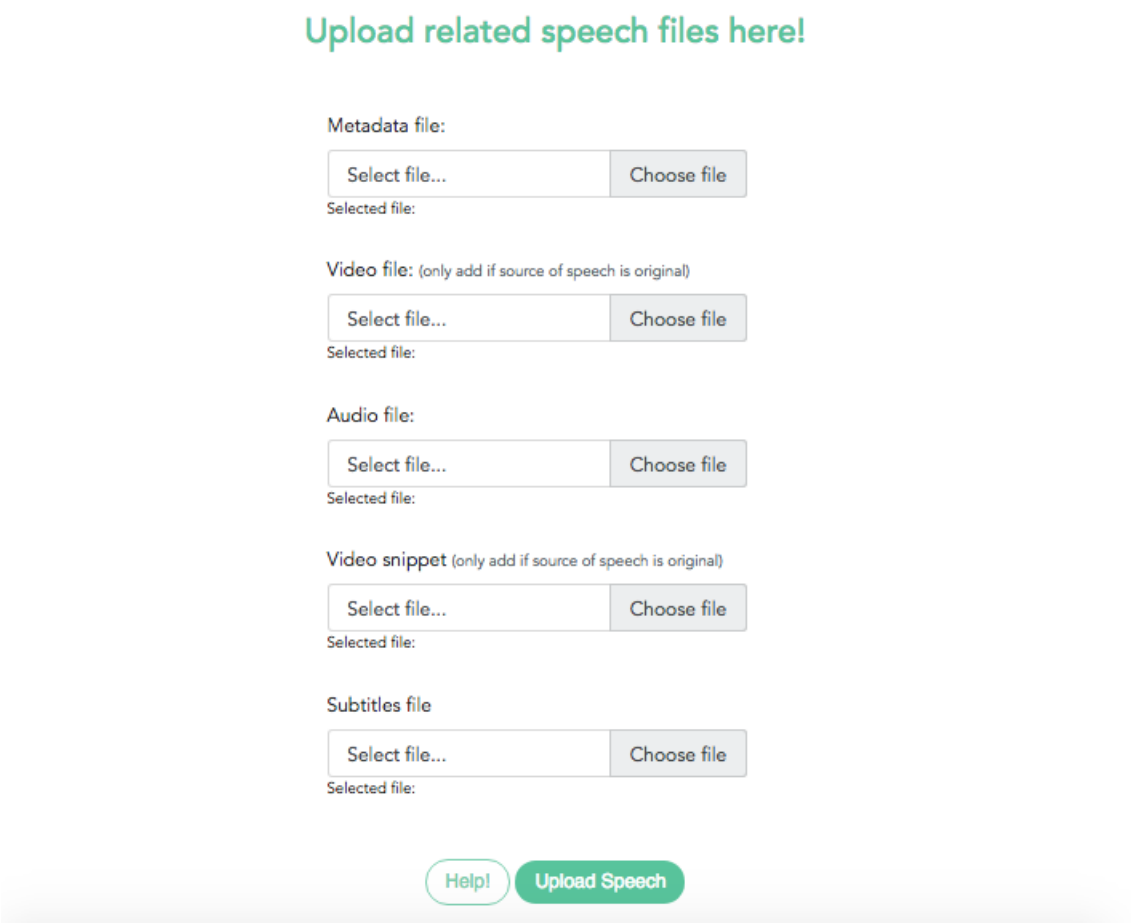


Figura 4.5: Componente *AddSpeech.vue*

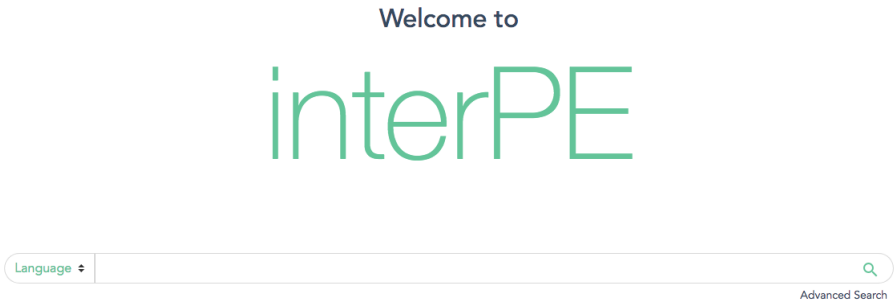


Figura 4.6: Componente *Home.vue*

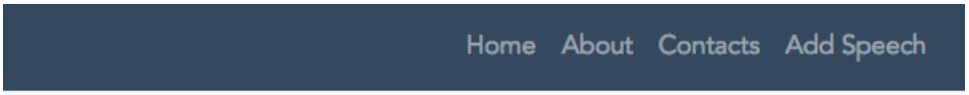


Figura 4.7: Componente *Nav.vue*

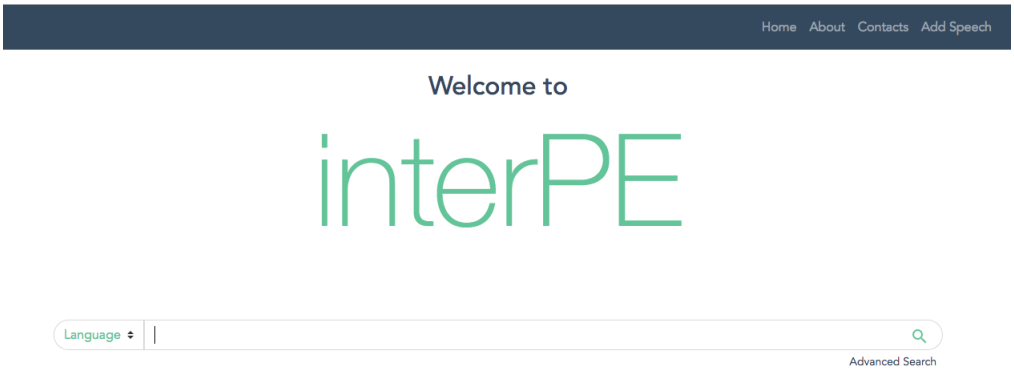


Figura 4.8: Componente *App.vue*

A hierarquia de todos os componentes constituintes da aplicação está representada na figura 4.9:

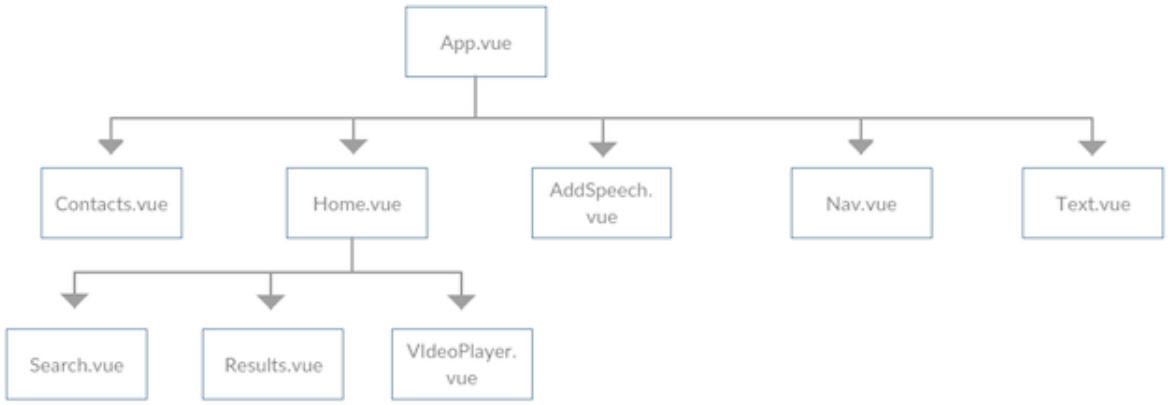


Figura 4.9: Hierarquia dos componentes do *interPE*

Todos os componentes mencionados foram construídos através de uma biblioteca chamada *BoostrapVue*; esta permite integrar todos os componentes do *Boostrap 4* com o *Vue.js*, assim, todo o processo de construção do *design* da interface do utilizador é facilitado e agilizado, uma vez que todos os componentes filhos já têm um estilo base e ainda, eventos já definidos.

Apesar de haver outras bibliotecas similares, a escolha recaiu sobre o *BootstrapVue* uma vez que era a única que integrava os componentes do *Bootstrap 4*, a versão mais actual do *Bootstrap*. Além disso, a sua documentação era simples e explícita, com uma *core team* sempre activa e ágil.

Por uma questão de simplificação do diagrama, e dado que todos os componentes presentes têm como filhos componentes importados do *BootstrapVue*, estes não se econtram representados na figura (4.9).

4.2.2 Pesquisar no InterPE

A funcionalidade de pesquisa envolve vários componentes, já citados no subcapítulo anterior, que são: *Home.vue (figura 4.6), Search.vue (figura 4.1), Results.vue (figura 4.2) e o Text.vue (figura 4.4).*

O objectivo da aplicação é ter uma pesquisa bilingue, ou seja, a pesquisa é realizada tanto nos discursos originais como nos interpretados numa dada língua, sendo que o foco do *interPE* é o português e inglês. Portanto, para executar uma pesquisa bastam dois simples passos: escolher a língua no menu *dropdown* no canto esquerdo da barra de pesquisa, e escrever, em concordância com a língua seleccionada, a palavra ou expressão que se pretende procurar. Caso o utilizador não escolha nenhuma língua, a aplicação não o deixa prosseguir com a pesquisa, enviando um alerta.

Se o utilizador desejar a realização de uma pesquisa mais completa, esta também é possível ao carregar no botão "*Advanced Search*"presente no canto inferior direito da barra de pesquisa (figura 4.1). Sempre que esta acção se verificar, é apresentado um *modal* (figura 4.10) com vários campos, entre eles: a possibilidade de pesquisar pelo género do orador, por data, pelo *SpeechID*, tópico, nome do orador e, ainda, pelo grupo político.

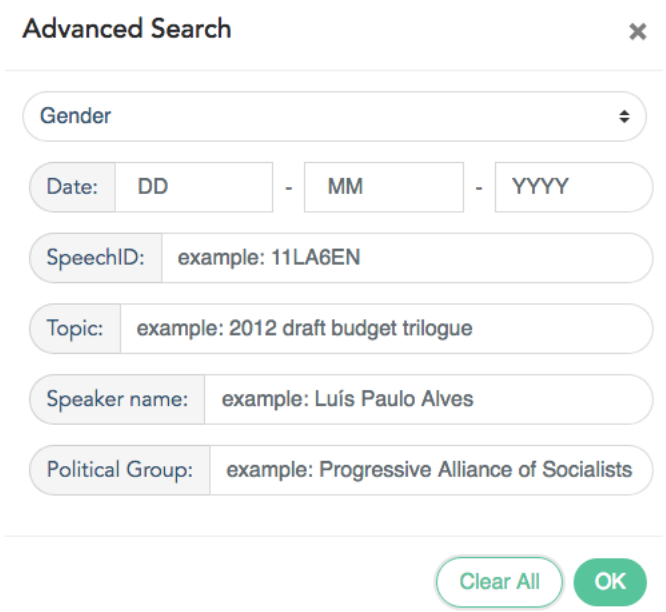


Figura 4.10: Opções de pesquisa avançada no *interPE*

Para a concretização da pesquisa avançada, basta preencher um dos campos disponíveis, não sendo obrigatório o preenchimento de todos. Ao clicar em "Ok", estando pelo menos um dos campos preenchidos, a pesquisa avançada fica activa e o botão com a opção de pesquisa avançada indica-o alterando o seu nome para "*Advanced Search is active*". Esta opção de pesquisa fica activa em todas as pesquisas realizadas pelo utilizador até que o mesmo a decida desactivar; para tal, basta voltar a carregar no botão de pesquisa avançada que apresenta o *modal* com os campos de pesquisa e carregar no botão *Clear all*; assim, a pesquisa volta ao seu funcionamento básico.

Após a realização da pesquisa, os dados inseridos pelo utilizador são enviados para o servidor, que, através de um *HTTP response*, envia os resultados obtidos. Uma vez que o *Vue.js* não tem integradas funcionalidades para pedidos AJAX (*Asynchronous JavaScript and XML*), a comunicação com o servidor é feita utilizando uma biblioteca externa. No caso do *interPE*, foi utilizada a biblioteca *Axios* [43], um cliente HTTP baseado em *promises* para o *browser* e para o *Node.js*. A escolha desta ferramenta recai sobre a familiariedade que havia com a mesma e na sua simplicidade.

Caso a pesquisa devolva resultados, estes são apresentados numa tabela (figura 4.2) com o *snippet* do vídeo do respectivo discurso, um excerto do discurso original e interpretado onde a

palavra ou expressão procurada está contida, sendo que esta aparece a negrito para dar maior destaque, e ainda, dois botões para aceder aos metadados do discurso interpretado e original e um para aceder às transcrições em texto (figura 4.4). Na tabela de resultados, o discurso original está sempre do lado esquerdo e o interpretado do direito, independentemente da fonte em que os resultados estejam a ser apresentados. Para alternar entre os resultados nos discursos original e interpretado, acima da tabela existem duas *tabs* que permitem visualizar as duas opções de resultados.

Todos os elementos da tabela que contêm os resultados são botões: o *snippet* do vídeo funciona como botão para a visualização do mesmo, com o respectivo áudio e texto (legenda) da fonte original e interpretada sincronizados, o próprio excerto das transcrições também serve como botão para, caso o utilizador não esteja interessado no discurso multimédia, aceder numa nova janela (abre um novo *tab*) ao texto do discurso original e interpretado, também é possível aceder a tal informação através do botão desginado para tal. Por fim, tal como já foi referido, existe um botão que permite consultar os metadados do discurso de cada fonte, apresentando um *modal* (figura 4.11) com as respectivas informações.

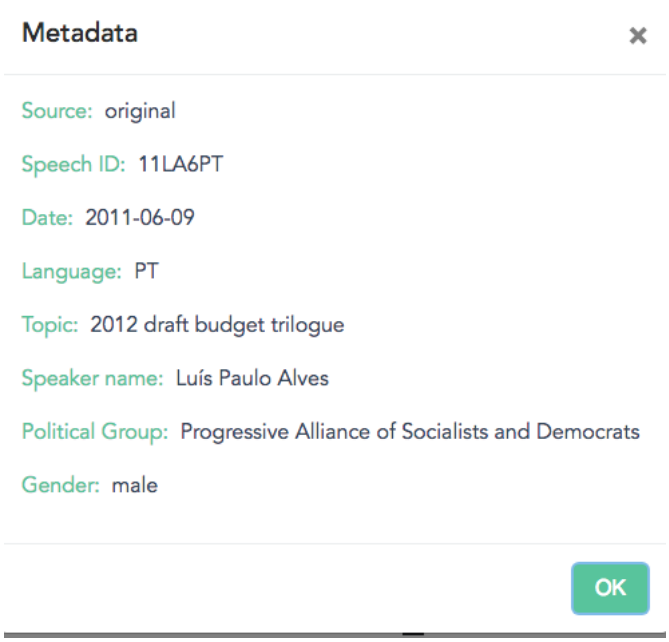


Figura 4.11: Consulta de metadados de um discurso no *interPE*

A opção de consultar apenas as transcrições dos discursos em texto, ao contrário de todas as outras opções, abre uma nova janela uma vez que, desta forma, o utilizador pode continuar

a consultar os resultados da pesquisa efectuada, sem a necessidade de repetir a mesma.

No caso de não serem retornados resultados de pesquisa, é apresentada uma mensagem a avisar o utilizador. Esta mensagem é apresentada com dois níveis de erro: o nível vermelho, para quando não há resultados tanto no caso da língua seleccionada ser a fonte original como interpretada; e o nível amarelo, quando apenas há resultados na pesquisa de fonte original ou interpretada.

Na figura 4.12, está representado o diagrama de sequências correspondente à execução de uma pesquisa, onde constam as acções descritas anteriormente.

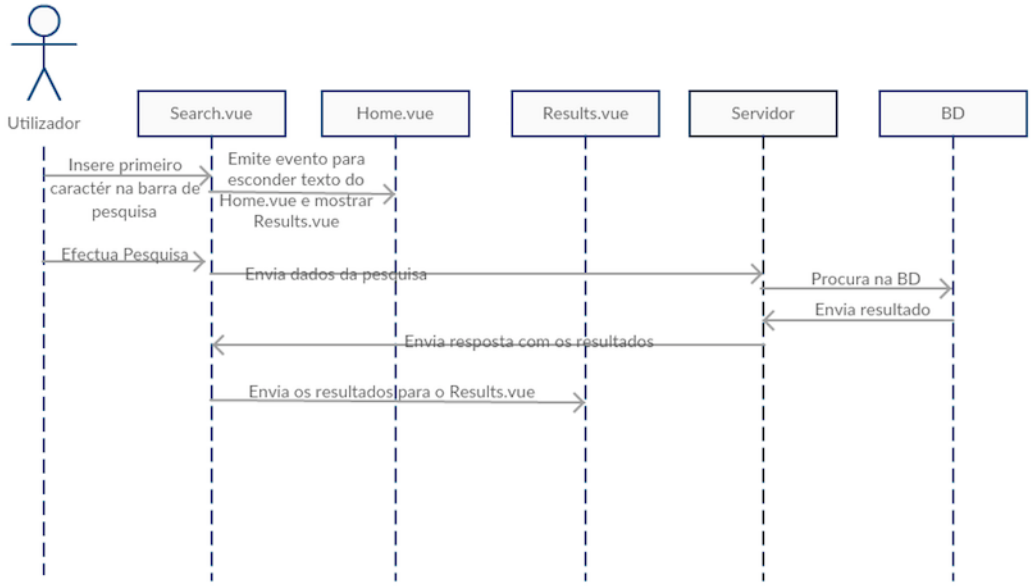


Figura 4.12: Diagrama de sequência de uma pesquisa no *interPE*

4.2.3 Visualização dos Discursos

Ao contrário da funcionalidade de pesquisa, que envolve vários componentes, esta apenas envolve o componente *VideoPlayer.vue*. Tal como podemos observar na figura 4.3, este componente é constituído por 3 elementos: o vídeo, o áudio e o texto.

Uma vez que, tal como referido no capítulo 'Estrutura da BD' (4.1), apenas a pista de áudio do vídeo difere entre o discurso de fonte original e interpretada, apenas é apresentado um vídeo, no entanto, o utilizador pode escolher qual a pista de áudio que pretende ouvir, estando esta sempre sincronizada com o vídeo. Por fim, também são apresentadas as legendas

(transcrições) original e interpretada lado a lado, sempre sincronizadas entre elas e com o áudio e vídeo.

Visto que um dos objectivos desta dissertação é a exploração de algumas das funcionalidades do HTML5, tanto para o áudio como para o vídeo, são utilizados os elementos *video* e *audio* de modo a embutir os mesmos na página *web*; assim, foi evitado o uso de *plugins* externos, como por exemplo, o *Flash Player*.

Para a realização do *interPE*, foram fornecidos os materiais de vídeo no formato '.mp4' e legendagem (transcrições) no formato '.exb'. Apesar do formato '.mp4' ser suportado pela maioria dos browsers utilizando a *tag video*, uma vez que este não estava codificado com o *codec h264*, a imagem do vídeo não era mostrada no caso do *Google Chrome*, sendo que este problema pode variar entre *browsers*. Para resolver este problema, foi utilizada uma ferramenta online para a conversão do vídeo no *codec* apropriado, chamada *Bear File Converter* [44].

No que toca ao formato das legendas em '.exb', este não é um formato de ficheiro de legendas, mas sim um ficheiro exportado do *software* EXMARaLDA [45], que inclui ferramentas de anotação e transcrição, gestão de *corpora* e análise de *queries*. Dado que a sua estrutura é tão díspar da estrutura de um ficheiro comum de legendas, e uma vez que não existem conversores de ficheiros '.exb' para este tipo, a opção foi manualmente criar um ficheiro '.srt', com a respectiva estrutura, baseada no conteúdo do ficheiro fornecido. A escolha do formato '.srt' prende-se com a sua simplicidade de estrutura, seguem-se duas figuras com o exemplo da estrutura de um ficheiro '.srt' (figura 4.12) e '.exb' (figura 4.13).

```
1
00:00:4.993 --> 00:00:7.192
Obrigado, Senhor Presidente.

2
00:00:7.194 --> 00:00:12.505
A política de coesão é a política-chave para o futuro do projecto europeu.

3
00:00:12.508 --> 00:00:24.065
A construção de um espaço comum, com um mercado interno, uma moeda própria, exige para poder
funcionar um determinado nível de convergência dos seus Estados-Membros e das suas regiões.

4
```

Figura 4.13: Estrutura de um ficheiro '.srt'

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) http://www.rrz.uni-hamburg.de/exmaralda -->
<basic-transcription>
<head>
<meta-information><project-name>11LA6PT</project-name><transcription-name>CISpt</transcription-name><referenced-file url="interv_2011-06-23_11_21_pt.wmv"/><
ud-meta-information><ud-information attribute-name="DATE">23-06-2011</ud-information><ud-information attribute-name="SOURCE">Original</ud-information></
ud-meta-information><comment></comment><transcription-convention>HIAT</transcription-convention></meta-information><speakertable><speaker id="SPK0">
<abbreviation>11LA6PT</abbreviation>
<sex value="m"/>
<languages-used><language lang="por"/></languages-used>
<l1><language lang="por"/></l1>
<l2></l2>
<ud-speaker-information>
<ud-information attribute-name="Political Group">SD</ud-information></ud-speaker-information>
<comment></comment>
</speaker>
</speakertable></head>
<basic-body>
<common-timeline>
<tli id="T0" time="0.0"/>
<tli id="T2" time="4.993053197808726"/>
<tli id="T3" time="7.192929773612302"/>
<tli id="T6" time="12.505965018810638"/>
<tli id="T11" time="24.065316480760345"/>
</common-timeline>
<tier id="TIE0" speaker="SPK0" category="v" type="t" display-name="11LA6PT [v]"><event start="T0" end="T2">*** </event>
<event start="T2" end="T3">Obrigado, Senhor Presidente. </event>
<event start="T3" end="T6">A política de coesão é a política-chave para o futuro do projecto europeu. </event>
<event start="T6" end="T11">A construção de um espaço comum, com um mercado interno, uma moeda própria, exige para poder funcionar um determinado nível de
convergência dos seus Estados-Membros e das suas regiões. </event>
</tier></basic-body>
</basic-transcription>
```

Figura 4.14: Estrutura de um ficheiro '.exb'

Após ultrapassados os problemas iniciais, o passo seguinte foi separar as pistas de áudio originais e interpretadas dos vídeos, de modo a poderem estar individualmente presentes na aplicação, utilizando o elemento *audio* do HTML5. Para tal, foi utilizado o *Online Video Converter* [46], uma ferramenta *online* que converte ficheiros vídeo, neste caso com a extensão ".mp4", para ficheiros áudio, cuja extensão seleccionada foi ".mp3".

Tal como já foi referido, todos os elementos (áudio, vídeo, texto) deste componente, *VideoPlayer.vue*, estão sincronizados entre si, ou seja, o áudio sincroniza o vídeo e, por sua vez, o vídeo sincroniza as transcrições. De modo a interagir com a reprodução do discurso, o utilizador pode apenas controlar o aúdio, uma vez que este é que irá conduzir a sincronização.

O HTML5 tem vários eventos associados aos elementos de áudio e vídeo, através dos quais se torna possível sincronizar a pista de áudio e vídeo dos discursos. Os eventos utilizados nesta aplicação são:

- *onplay*: quando o vídeo começa a reproduzir [47];
- *onseeked*: quando o utilizador termina de retroceder ou avançar para uma nova posição [48];

- *onpause*: quando o vídeo é pausado [49];

Portanto, o vídeo e áudio do discurso mantêm-se sincronizados pois, assim que um dos eventos acima é accionado pelo elemento *audio*, o instante de reprodução do vídeo é alterado para o instante a que o áudio está a ser reproduzido. Uma vez que há duas pistas de áudio a controlar o vídeo, este só fica em pausa se o evento *onpause* tiver sido emitido pelas duas faixas de áudio, senão, o vídeo continua sincronizado com a faixa de áudio que está a ser reproduzida. O vídeo será sempre sincronizado com o instante de tempo emitido pelo evento mais recente. De notar ainda que o áudio foi escolhido como fonte de controlo por parte do utilizador uma vez que assim, este consegue controlar cada fonte, original ou interpretada, do discurso separadamente.

Por fim, a sincronização entre o vídeo e as legendas é feita através de um temporizador que de 10 em 10 milisegundos, verifica o instante de reprodução do vídeo, compara esse instante com o instante de reprodução das legendas e mostra a legenda correspodente ao instante de tempo correcto. Por uma questão de contexto, para além da frase que está a ser referida no momento, também são mostradas as duas falas anteriores e as duas seguintes, sendo que a fala presente é apresentada com destaque. Estas frases estão contidas em linhas de uma tabela HTML.

É importante também referir que, ainda na fase de apresentação de resultados, quando o utilizador selecciona um discurso, este irá começar a sua reprodução no instante em que a frase presente nos resultados é mencionada. Para além disso, a pista de áudio que é reproduzida por defeito é a correspondente ao discurso original.

Por fim, na figura 4.15 pode-se observar o diagrama de sequência da acção de visualização de um discurso multimédia.

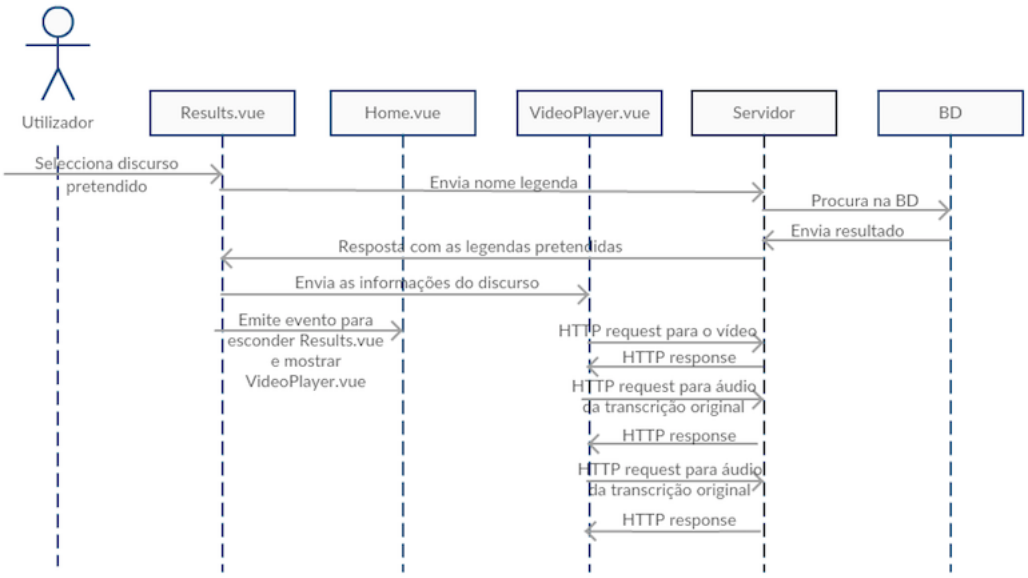


Figura 4.15: Diagrama de sequência da função de visualização de discursos do *interPE*

4.2.4 Adição de Discursos

A adição de discursos tem como objectivo o contínuo crescimento do *interPE*, sem a necessidade do acesso directo à BD, assim, os utilizadores devidamente creditados para tal, poderão continuar a manter esta plataforma actualizada.

Tal como no caso anterior de visualização de discursos, esta funcionalidade apenas envolve um componente, o *AddSpeech.vue* (figura 4.9), que é constituído por um formulário com vários campos para adicionar os ficheiros necessários à criação de um novo documento que representa um discurso, no *MongoDB*.

Tal como referido na secção Estrutura da Base de Dados (4.1), cada documento da colecção 'vídeo' é constituído por vários campos. Neste componente, encontramos os *inputs* para o utilizador inserir os ficheiros respectivos a um discurso na colecção "vídeo"do *MongoDB*. No caso da fonte do discurso ser interpretada, os campos para inserir o ficheiro de vídeo, o snippet e ainda, o campo *political Group* e *speaker name* do objecto que contém os metadados, não deverão ser preenchidos. Após a submissão dos ficheiros, todos são enviados para o servidor através da utilização do *Axios*, tal como referido na secção Pesquisar no InterPE.

O primeiro passo para adicionar um novo discurso é o preenchimento de um formulário

de *login* (figura 4.16). Ora, apenas o gestor do *interPE* pode adicionar novos discursos à plataforma.

Na secção "Estrutura da Base de Dados"(4.1), não é mencionada nenhuma colecção responsável por armazenar as informações de *login* dos utilizadores; isto deve-se ao facto de ter sido utilizado o *Auth0* [50], uma plataforma que oferece como serviço a autenticação e autorização em aplicações *web, mobile* ou *desktop*, para a implementação de tal funcionalidade.

A decisão de adopção do *Auth0* foi tomada uma vez que seria mais rápido e fácil de implementar a autenticação desta forma do que construindo todas as funcionalidades de raiz, tal como é referido no *website* da plataforma [50]. Com o *Auth0*, é possível implementar a autenticação utilizando o *Lock widget*, um formulário de *login* fornecido pela plataforma, ou utilizando o nosso próprio formulário. Por uma questão de estética, para manter o estilo da restante aplicação, foi decidida a segunda hipótese.

Uma vez que o *Auth0*, no caso de aplicações *web* que utilizem o *Vue.js*, apenas disponibiliza um guia para a implementação da autenticação utilizando o *Lock Widget* e, visto que após uma pesquisa todos os exemplos encontrados que não utilizam o *Lock widget* se encontram numa versão anterior, a implementação da autenticação utilizando esta ferramenta acabou por ser muito morosa. Ou seja, tanto a facilidade como a rapidez de implementação, que foram as vantagens que inicialmente levaram a esta decisão, não se cumpriram devido à confusa documentação do *Auth0*.

Após o *login* (figura 4.16) ter sido executado correctamente, o gestor acede finalmente ao componente responsável pela adição de novos discursos. Para além disso, existe ainda um botão de ajuda ("*Help*") para o caso de existir alguma dúvida quanto ao formato dos ficheiros, ou outros cuidados a ter ao adicionar um novo discurso no *interPE*. O botão referido abre um *modal* (figura 4.17) com as várias informações.

Caso o utilizador tente inserir um ficheiro com um nome já existente, é apresentada uma mensagem de erro e o discurso não é inserido na BD.

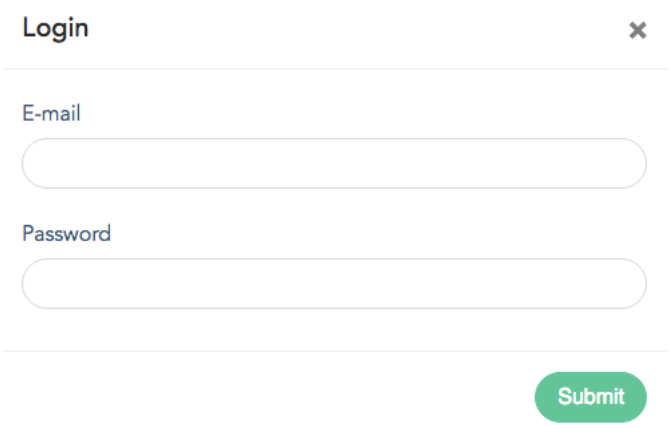


Figura 4.16: Formulário de login do *interPE*

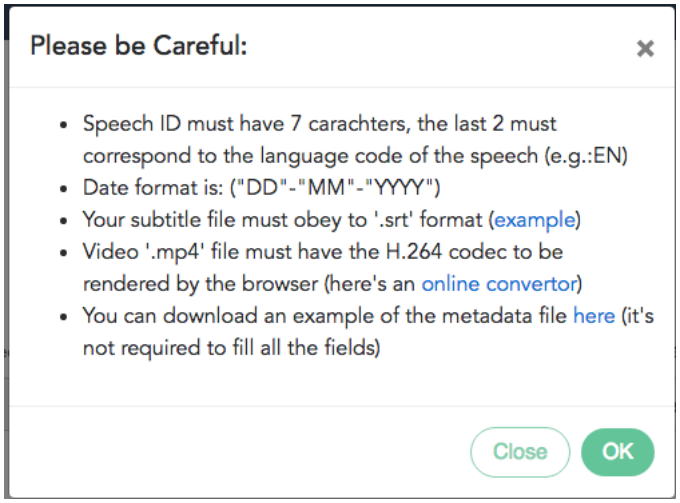


Figura 4.17: *Modal* de ajuda na adição de um novo discurso no *interPE*

A figura 4.18 representa o diagrama de sequência correspondente à adição de um novo discurso por parte do utilizador.

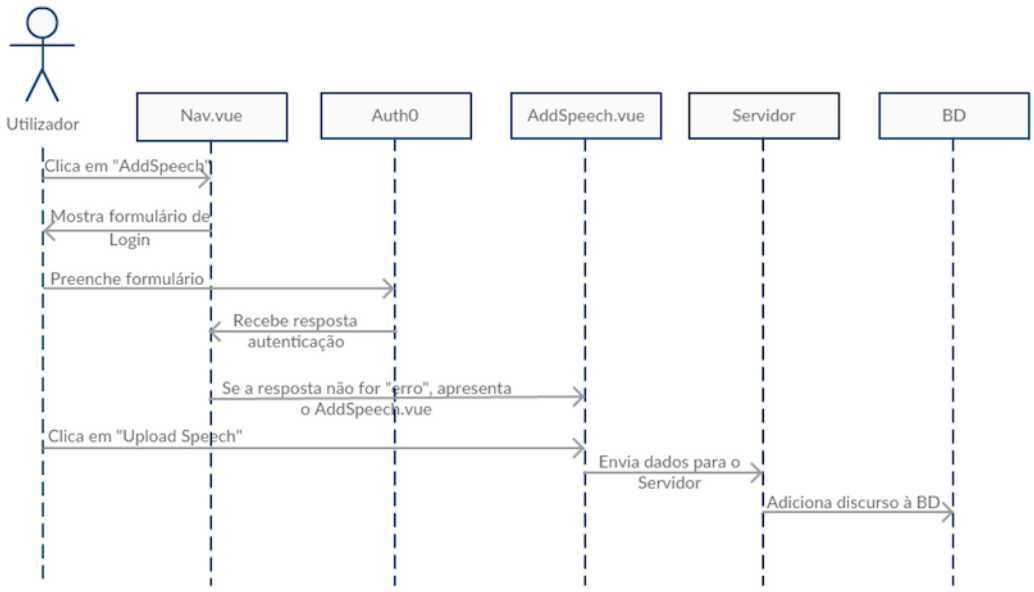


Figura 4.18: Diagrama de sequência da função de adição de discursos do *interPE*

4.3 Servidor

Esta secção trata o lado do servidor da aplicação desenvolvida, explorando a estrutura do mesmo e as suas funcionalidades. O desenvolvimento deste foi realizado utilizando *Node.js* [22], uma moderna plataforma de execução de Javascript do lado do servidor e amplamente utilizada.

Para além disso, a utilização de *Node.js* oferece grandes vantagens, tais como a utilização de um modelo de programação assíncrono, ou seja, a execução do código não é bloqueada, tornando o desempenho da aplicação mais rápido e ainda, o facto deste ser *JavaScript*, portanto, esta linguagem é utilizada tanto no *frontend* como no *backend*.

Os próximos subcapítulos abordam as funcionalidades da aplicação no lado do *backend*.

4.3.1 Estrutura Servidor

O servidor está organizado de modo a que haja uma separação de conceitos, isto é, em vez de todo o código estar presente num só ficheiro, o que leva a uma díficil manutenção do mesmo, as várias secções de código são agrupadas consoante a sua função.

A estrutura do servidor consiste no seguinte:

- *Controllers*: os ficheiros com as funções responsáveis por lidar com os pedidos HTTP (*Hypertext Transfer Protocol*): estes recebem a informação do cliente e devolvem uma resposta HTTP;
- *Routes*: onde são armazenados os ficheiros responsáveis pelo encaminhamento dos pedidos do *website* para os *controllers*, ou seja, consoante a URL (*Uniform Resource Locator*) do pedido HTTP, estes encaminham-no para a função apropriada para tratar o mesmo;
- *Models*: ficheiros que contêm os modelos para interacção com a BD;
- *server.js*: onde é iniciado o servidor e a conexão com a Base de Dados;
- *media*: onde todos os ficheiros associados aos discursos são armazenados.

Os ficheiros *controllers*, *models* e *routes* estão associados à colecção da BD com que interagem, portanto, uma vez que apenas existe uma colecção, "vídeo", na BD, apenas existe um ficheiro *controller*, *videoController.js*, um ficheiro *route*, *videoRoute.js*, e um ficheiro *model*, *videoModel.js*.

4.3.2 Pesquisar e Visualizar Discursos no *interPE*

No subcapítulo Cliente (4.2), já foi abordada a funcionalidade de pesquisa no *frontend*, aqui é abordada a mesma funcionalidade mas do lado do *backend*.

Desde o momento em que o utilizador efectua uma pesquisa, até à visualização dos resultados, o servidor executa várias acções. Quando é efectuada uma pesquisa, o servidor recebe um um *HTTP request* do cliente, que contém um objecto com os seguintes campos:

- *query*: a palavra ou expressão inserida pelo utilizado na barra de pesquisa;
- língua: a língua seleccionada pelo utilizador;
- *advanced*: objecto que contém todos os campos da pesquisa avançada;

Tanto para o caso da pesquisa simples como para o caso da pesquisa avançada, o objecto *advanced* é enviado, no entanto, se o modo de pesquisa for simples, este tem todos os seus campos nulos. Já no caso do modo de pesquisa avançada, o objecto *advanced* contém a

informação preenchida pelo utilizador nos seus respectivos campos, não sendo obrigatório que todos os campos sejam preenchidos. Os que não são preenchidos, tal como no caso da pesquisa simples, são enviados nulos.

Após recebida esta informação, o servidor executa duas funções em paralelo. Ambas têm como objectivo percorrer a BD à procura de alguma correspondência de acordo com as informações inseridas pelo utilizador, no entanto, uma procura nos discursos originais cuja lingua corresponde à seleccionada e a outra procura nos discursos em que a língua do discurso interpretado é correspondente à seleccionada. No caso de existência de resultados, ambas devolvem um objecto que contém o discurso original e a interpretação correspondente.

De modo a implementar a pesquisa na Base de Dados, é necessária a criação de uma conexão entre o servidor e a mesma. Para tal, foi utilizado o *Mongoose* que é um *Object Document Mapper* (ODM), ou seja, este é responsável por criar um mapeamento entre um documento da BD e um *object model*, uma representação dos documentos definidos no *MongoDB*. Uma vez que na BD apenas foi definida uma única colecção 'vídeo', tal como referido do subcapítulo "Escolha da Base de Dados"(3.5.2), então, no servidor também foi apenas definido um único modelo correspondente a essa colecção de documentos. A pesquisa é feita utilizando a funcionalidade de *Text Search* do *MongoDB*, que permite realizar pesquisas de texto no conteúdo de *strings*.

Após obtidos os resultados, estes são enviados para o cliente como resposta ao pedido HTTP recebido. Os *HTTP requests* do lado do servidor são geridos pelo *Express.js* [51], a *framework* padrão do *Node.js* que permite, entre outros, receber pedidos HTTP realizados ao servidor e enviar a respectiva resposta.

Tal como referido no subcapítulo 4.2 (Cliente), após a visualização dos resultados, o utilizador tem várias opções, entre elas a visualização multimédia do discurso seleccionado e a visualização do mesmo em formato de texto. No caso da visualização multimédia do discurso, são enviados pedidos HTTP para o servidor com o nome dos ficheiros de áudio e vídeo e como resposta este envia os ficheiros designados, tal como representado na figura 4.15, do subcapítulo 4.2. É também enviado um pedido com as legendas pretendidas, o servidor devolve as mesmas no formato de *array*.

Por fim, para o caso da visualização dos discursos em formato texto, são enviados dois pedidos, um para obter a transcrição do discurso original e outra para a obtenção da transcrição

do discurso interpretado. As legendas de ambos os pedidos são enviados em *string* para o cliente.

4.3.3 Adição de Discursos

O processo de adição de novos discursos do lado do servidor é aqui descrita, sendo bastante simples.

Após o utilizador submeter todos os ficheiros referentes a um discurso estes são enviados para o servidor, que verifica se não existem ficheiros com o mesmo nome dos inseridos. Caso não existam ficheiros com o mesmo nome, este cria um objecto com os dados recebidos e insere-o como um novo documento no *MongoDB*. Tal como referido no subcapítulo "Estrutura do servidor"(4.3.1), existe uma pasta *media* onde todos os ficheiros adicionados são guardados. Caso já existam ficheiros com o mesmo nome, o servidor envia uma mensagem de erro para o cliente.

No caso dos ficheiros que contêm os metadados e as legendas, estes não são guardados directamente na pasta referida anteriormente. Primeiro, são integralmente lidos pelo servidor, depois, no caso das legendas, estas são guardadas numa *string*. Já no caso do ficheiro referente aos metadados, as suas informações são guardados num objecto.

Por fim, o nome dos ficheiros guardados na pasta *media*, assim como a *string* que contém as legendas e, ainda, o objecto com os metadados são inseridos nos respectivos campos, criando um novo documento da colecção 'vídeo' da Base de Dados.

5 Conclusão

5.1 Resultados

De modo a testar se os requistos do sistema foram cumpridos correctamente, foram realizados alguns testes. Primeiro, para entender se realmente a experiência de utilização do *interPE* resultava em algo simples e intuitivo para o utilizador, quer com conhecimentos na àrea da Linguística, que já interagiu com outros *corpora* multimédia, quer sem conhecimentos nesta área, foi dada a testar a aplicação aos dois tipos distintos de utilizador. A observação retirada é que ambos os casos conseguiram manusear a aplicação sem qualquer explicação de funcionamento prévia, no entanto, para os utilizadores sem conhecimentos na área, foi necessária a explicação do conceito de corpus linguístico de modo a entender o objectivo da aplicação. Ambos realizaram testes utilizando o modo de pesquisa simples e avançado.

Em segundo lugar, uma vez que a função de adição de novos discursos apenas é executável pelo administrador e, uma vez que este se insere no grupo de utilizadores com conhecimentos na área da Linguística, esta opção apenas foi dada a testar a este tipo de utilizadores, não tendo sido encontradas dificuldades na adição dos ficheiros, sendo que a aplicação valida os formatos dos ficheiros a serem adicionados, não permitindo que formatos inadequados sejam adicionados, consoante o tipo de dados a inserir.

A aplicação foi testada utilizando um *Macbook Pro* de 13 polegadas, em dois *browsers* diferentes, o *Google Chrome* e o *Safari*, não tendo sido apresentados problemas na interface do *interPE* para ambos os casos.

5.2 Conclusões

O objectivo desta dissertação era a construção de uma aplicação *web* que albergasse o *corpus* audiovisual do Parlamento Europeu, fornecendo uma interface simples e intuitiva. A

interface deveria oferecer aos seus utilizadores dois modos de pesquisa, simples e avançada, e ainda, a opção de adição de novos discursos ao *corpus*. Para além disso, a aplicação deveria ser uma *Single Page Application*, levando à exploração de novas funcionalidades oferecidas pelo HTML5.

O objectivo da criação de uma aplicação de fácil utilização foi cumprido pois, tal como discutido no subcapítulo anterior, tanto os utilizadores sem conhecimento na área da linguísta, como os que têm conhecimentos nesta área conseguiram utilizar o *interPE* sem qualquer dificuldade.

Este projecto foi desenvolvido utilizando como *frontend framework* o *Vue.js*, assim, através da construção de vários componentes, foi possível atingir o comportamento de uma SPA. Também foi tirado partido de novas funcionalidades do HTML5 no que toca à reprodução de vídeo e áudio numa página *web*.

Para além disso, o servidor construído oferece suporte para todos os dados utilizados: uma BD que segue um modelo de dados adequado ao problema, nomeadamente não relacional; e um ambiente de execução JavaScript moderno e assíncrono, que permite a exploração simples do modelo de processamento paralelo para tirar partido das máquina com vários processadores e melhorar a resposta da aplicação.

Concluindo, os resultados obtidos mostram que os objectivos inicialmente propostos foram cumpridos, resultando numa aplicação SPA, o *interPE*, que oferece diferentes modos de pesquisa e, ainda, a manutenção e actualização do *corpus*, mantendo uma interface simples, atractiva e intuitiva.

5.3 Trabalho Futuro

Apesar do *interPE* ter cumprido os requisitos propostos inicialmente, ainda há espaço para melhorar e acrescentar funções ao mesmo. Em termos de *corpus* multimédia, este ainda oferece um nível de funcionalidades básico, pelo que a adição de, por exemplo, a capacidade de anotação morfossintáctica e de fenómenos da oralidade, como por exemplo, hesitações, reformulações ou truncamentos, seria uma mais valia para o mesmo.

No caso das funções de administrador, para além da capacidade de adição de vídeos, seria interessante e útil a possibilidade de listar todos os discursos que constam no *corpus*, assim como a possibilidade de eliminar ou editar os mesmos. Isto traria vantagens à manutenção

da plataforma, tornando-a mais simples e organizada.

A opção de consulta dos dados referentes a um discurso apenas está presente aquando da apresentação dos resultados. Seria também útil que fosse possível consultar os metadados de um discurso na página de visualização do mesmo. Para além disso, ao visualizar um discurso, para além do controlo do mesmo através das faixas de áudio, também poderia ser oferecida a possibilidade de ao carregar numa das frases das legendas, o vídeo e áudio retrocedessem ou avançassem para o momento em que a mesma foi proferida.

Por último, outra funcionalidade interessante seria a manipulação da velocidade da faixa de áudio do discurso pois, assim, o processo de formação de intérpretes poderia ser facilitado, uma vez que estes, num momento incial, poderiam optar por discursos com uma velocidade menor e, consoante o seu avanço na formação, poderiam gradualmente aumentar a velocidade dos discursos. Para além disso, esta funcionalidade também traria vantagens no processo de ensino e aprendizagem de línguas, uma vez que os alunos com menor nível de proficiência linguística poderiam treinar as suas competências de compreensão oral, processando discursos menos rápidos numa fase inicial da aprendizagem.

Referências

[1] "Definição de Corpus no Dicionário Priberam". https://www.priberam.pt/dlpo/corpus.

[2] "European Parliament Interpreting Corpus". http://sslmitdev-online.sslmit.unibo.it/ corpora/corporaproject.php?path=E.P.I.C.. 2004.

[3] Arkhangel'skii, T. A. e Sozinova, O. A. "A multimedia corpus of the Yiddish language". 2015.

[4] "Definição de Iídiche no Dicionário Priberam". https://www.priberam.pt/dlpo/i%C3%ADdiche.

[5] "Child Language Data Exchange System". http://childes.talkbank.org. 2003.

[6] University of Glasgow School of Critical Studies. "Scottish Corpus Of Texts & Speech". 2002.

[7] Hasebe, Yoichiro. "TED corpus search engine". https://yohasebe.com/tcse/. 2015.

[8] Myles, F. e Mitchell, R. "French Learner Language Oral Corpora". http://www.flloc.soton.ac.uk/. 2009.

[9] "Spanish Learner Language Oral Corpora". http://www.splloc.soton.ac.uk/search.php. 2006.

[10] Mitchell, R. Marsden, E. e Myles, F. "Linguistic Development in L2 Spanish: Creation and analysis of a learner corpus". 2008.

[11] Dios, Patricia Sotelo e Guinovart, Xavier Gomez. "A multimedia parallel corpus of english-galician film subtitling". Em Alberto Simões, Ricardo Queiros Daniela da Cruz (eds.), st symposium on languages, applications and technologies, 2012, 255-266.

[12] Hasebe, Yoichiro. "Design and Implementation of an Online Corpus of Presentation Transcripts of TED Talks". http://www.sciencedirect.com/science/article/pii/S1877042815044353. 2015.

[13] "Definição de Lema no Dicionário Priberam". https://www.priberam.pt/dlpo/lema.

[14] "Corpus Paralelo CLUVI". http://sli.uvigo.gal/CLUVI/info.html. 2003.

[15] Bendazzoli, Claudio e Annalisa Sandrelli. "An approach to corpus-based interpreting studies: Developing EPIC (European Parliament Interpreting Corpus)". 2005.

[16] W3C HTML Working Group. "HTML5, A vocabulary and associated APIs for HTML and XHTML". https://www.w3.org/TR/html5/. 2014.

[17] MDN web docs. "Media formats for HTML audio and video". https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats.

[18] W3Schools. "HTML5 Audio". https://www.w3schools.com/html/html5_audio.asp.

[19] W3Schools. "HTMl5 Video". https://www.w3schools.com/html/html5_video.asp.

[20] W3C. "Cascading Style Sheets". https://www.w3.org/Style/CSS/. 2017.

[21] MDN web docs. "JavaScript". https://developer.mozilla.org/pt-PT/docs/Web/JavaScript. 2015.

[22] Node.js Foundation. "Node.js". https://nodejs.org/en/. 2017.

[23] "Common.js". http://www.commonjs.org/. 2009.

[24] MongoDB. "NoSQL Explained". https://www.mongodb.com/nosql-explained.

[25] MongoDB. "What is a non relational Database". https://www.mongodb.com/scale/what-is-a-non-relational-database.

[26] "JSON". http://www.json.org/. 1999.

[27] Wikipedia. "JSON". https://pt.wikipedia.org/wiki/JSON. 2017.

[28] W3. "Hypertext Transfer Protocol". https://www.w3.org/Protocols/rfc2616/rfc2616.html. 2004.

[29] Mikhailov, M. e Cooper, R. "Corpus Linguistics for Translation and Contrastive Studies: A Guide for Research". https://books.google.pt/books?id=1PgyDAAAQBAJ. 2016, Routledge Corpus Linguistics Guides, Taylor & Francis.

[30] You, Evan. "Vue.js". https://vuejs.org. 2014.

[31] Vue.js. "The Vue Instance". https://vuejs.org/v2/guide/instance.html.

[32] "JavaScript Frameworks Benchmark". https://rawgit.com/krausest/js-framework-benchmark/master/webdriver-ts/table.html.

[33] CodersEye. "11 Best PHP Frameworks for Modern Web Developers in 2017". https://coderseye.com/best-php-frameworks-for-web-developers/. 2017.

[34] Wikipedia. "Database". https://en.wikipedia.org/wiki/Database. 2017.

[35] Wikipedia. "Relational Database". https://en.wikipedia.org/wiki/Relational_database. 2017.

[36] Grow the Future. "A General Overview of Relational vs Non Relational Databases". http://growthefuturenow.com/relational-vs-non-relational-databases/.

[37] Tech Republic. "10 things you should know about NoSQL databases". http://www.techrepublic.com/blog/10-things/10-things-you-should-know-about-nosql-databases/.

[38] "MongoDB". https://www.mongodb.com.

[39] "BSON". http://bsonspec.org.

[40] MongoDB. "JSON and BSON". https://www.mongodb.com/json-and-bson.

[41] Wikipedia. "Single-page application". https://en.wikipedia.org/wiki/Single-page_application#Client.2FServer_code_partitioning. 2017.

[42] Vue.js. "What are components". https://vuejs.org/v2/guide/components.html#What-are-Components.

[43] "Axios". https://github.com/mzabriskie/axios.

[44] "Bear File Converter". https://www.ofoct.com/video-converter/convert-to-h-264-video.html.

[45] "EXMARaLDA". http://exmaralda.org/en/.

[46] "Online Video Converter". https://www.onlinevideoconverter.com/pt.

[47] w3schools. "HTML Audio/Video DOM play Event". https://www.w3schools.com/tags/av_event_play.asp.

[48] w3schools. "onseeked Event". https://www.w3schools.com/jsref/event_onseeked.asp.

[49] w3schools. "HTML Audio/Video DOM pause Event". https://www.w3schools.com/tags/av_event_pause.asp.

[50] "Auth0". https://auth0.com/.

[51] Express.js. https://en.wikipedia.org/wiki/Express.js.